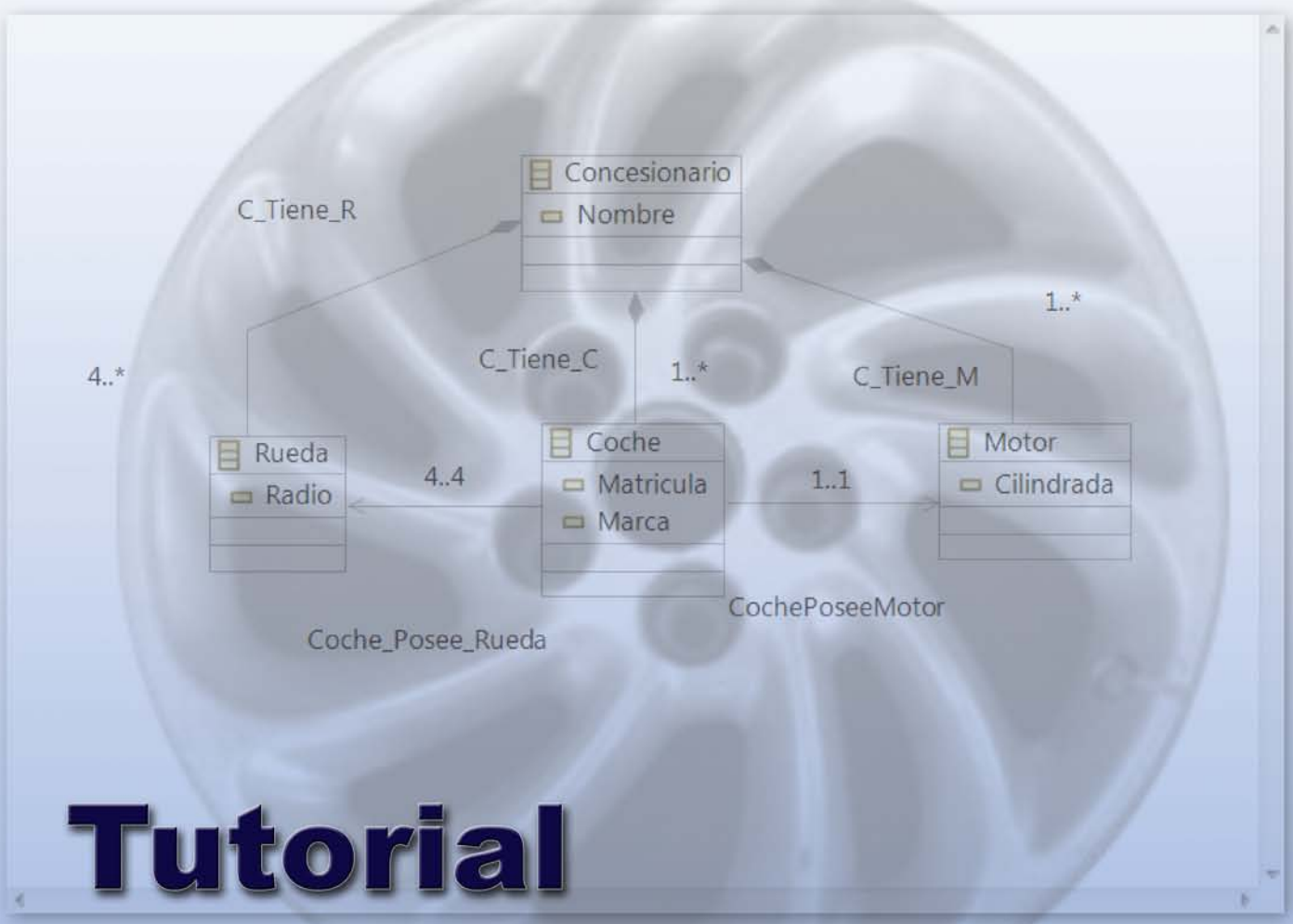


Tutorial de introducción a EMF y GMF

DEFINICIÓN DE LENGUAJES ESPECÍFICOS DE DOMINIO

DAVID MUSAT SALVADOR
dmusat@syst.eui.upm.es
JENNIFER PÉREZ BENEDÍ
jenifer.perez@eui.upm.es
PEDRO PABLO ALARCÓN CAVERO
pcavero@eui.upm.es

POLITÉCNICA

ÍNDICE

1. INTRODUCCIÓN

Este tutorial tiene como objetivo proporcionar las nociones básicas para crear herramientas de modelado específicas de dominio utilizando las herramientas de modelado *Eclipse Modeling Framework (EMF)* [EMF09] y *Graphical Modeling Framework (GMF)* [GMF09]. Ambas proporcionadas por la plataforma Eclipse [ECL09].

Antes de comenzar el tutorial se detallan los requisitos mínimos de instalación para que EMF y GMF se ejecuten correctamente. Para ello, se proporciona una pequeña guía sobre cómo instalar eclipse y las librerías necesarias para la ejecución de estas aplicaciones.

Finalmente, cabe destacar que a lo largo del tutorial se utilizará un sencillo caso de estudio para ilustrar al lector cada uno de los pasos. Dicho caso de estudio también se introduce antes de dar paso al tutorial.

2. REQUISITOS MÍNIMOS

Para dar comienzo a este tutorial es necesario disponer de un ordenador con al menos 256 MB de memoria RAM. Por otro lado, el software que se ha de tener instalado es el siguiente:

- Eclipse build S-3.2M5.
- JDK 1.5
- EMF 2.2.0M5.
- GEF 3.2M5.
- GMF 1.0M5.

A continuación, se presenta brevemente la plataforma Eclipse y se muestra al lector como descargar e instalar la versión de Eclipse que incluye las herramientas EMF, GEF (Graphical Editing Framework) y GMF.

2.1 Descripción de Eclipse

Eclipse es una herramienta que permite integrar diferentes aplicaciones para construir un entorno integrado de desarrollo (IDE(Integrated Development Environment)). Es una plataforma de desarrollo de software abierto (*open-source*), que está dividida en tres partes: Eclipse Project, Eclipse Tools Project y Eclipse Technology Project.

El componente principal de Eclipse Project es JDT (Java Development Tools). Éste permite crear aplicaciones en Java. Además, Eclipse proporciona mecanismos para integrar otras aplicaciones en forma de *plug-ins*. Estos plug-ins son reconocidos automáticamente por Eclipse cuando se inicia. Dado que Eclipse está desarrollado en Java, para su funcionamiento se debe tener instalado el JRE (Java Runtime Enviroment). Una vez instalado, Eclipse detecta automáticamente la ubicación del JRE.

En la actualidad existe una fuente principal de documentación y descarga de aplicaciones y complementos en Internet sobre la plataforma Eclipse. En este sitio es posible encontrar recursos, documentos, tutoriales, "wikis", comunidades de usuarios, referencias a proyectos relacionados, etc.

http://www.eclipse.org/

2.2 Instalación de Eclipse

La plataforma de desarrollo Eclipse es *open-source* y la web del proyecto Eclipse facilita la descarga del software que hay que instalarse. La dirección web de la que se puede descargar la plataforma de desarrollo es la siguiente:

http://www.eclipse.org/downloads/

Una vez se ha accedido a dicha web, de entre todas las diferentes versiones estables de Eclipse que proporciona, hay que descargarse la versión *Eclipse Modeling Tools (includes Incubating components)*. Esta versión incluye todos los recursos especificados en los requisitos necesarios para realizar este tutorial.

Una vez seleccionada esta versión, la página web de la versión elegida nos proporciona en su parte superior la lista de servidores que permiten descargar dicha versión. Se deberá elegir uno de los servidores que aparecen en la lista desplegable para comenzar la descarga.

A continuación, es posible proceder a la instalación del entorno Eclipse. Para ello, una vez finalizada la descarga deberá descomprimirse el fichero descargado en la carpeta que se desee del disco, por ejemplo, C:\eclipse. En esta carpeta estará el acceso directo que lanza a ejecución la plataforma de desarrollo Eclipse.

2.3 Ejecución de Eclipse

Antes de comenzar a trabajar con Eclipse es necesario crearse un espacio de trabajo donde guardar los proyectos que se realicen. En el caso de este tutorial, el caso de estudio utilizado versa sobre un concesionario de automóviles, por lo que vamos a crear en disco una carpeta de nombre *Concesionario* donde guardar el desarrollo del caso de estudio. Posteriormente, se lanza a ejecución la plataforma Eclipse, pidiéndonos que seleccionemos un espacio de trabajo (*workspace*). En realidad un espacio de trabajo simplemente establece en qué carpeta de disco se almacenarán los diferentes ficheros que compongan un proyecto en Eclipse. En este caso, el espacio de trabajo corresponderá con la carpeta *Concesionario* (ver Figura 1).

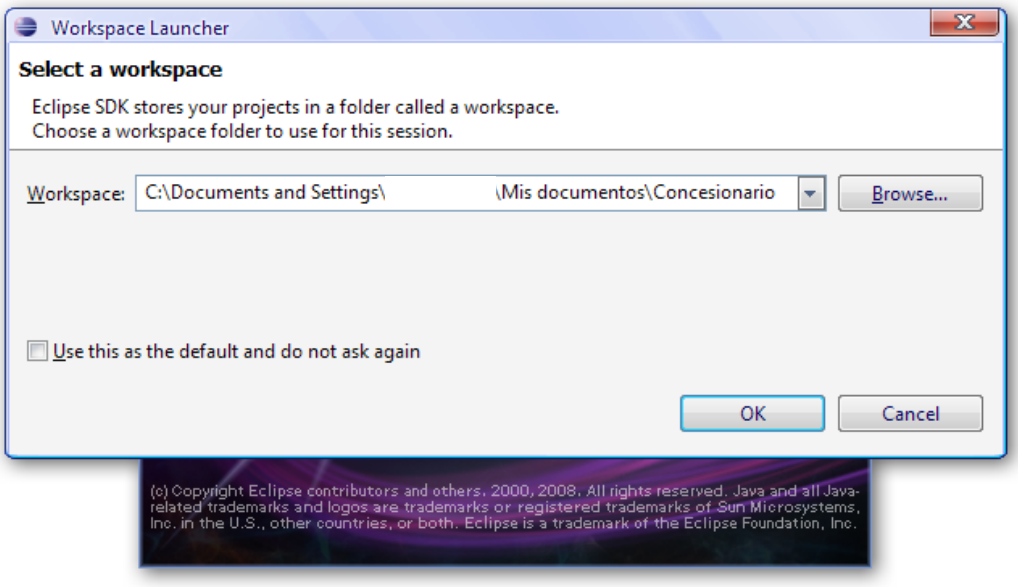


Figura 1: Selección del WorkSpace

Una vez abierto el Workspace, con el objetivo de comprobar que nuestra instalación de Eclipse tiene todas las librerías necesarias para utilizar EMF y GMF, hacemos click en *Help → About* Eclipse SDK y pulsamos en el botón de *Plug-in details* (ver Figura 2). Una vez realizado esto, la lista de plug-ins instalados se muestra en una ventana, pudiendo comprobar que los plug-ins de GMF y EMF están correctamente instalados (ver Figura 3).

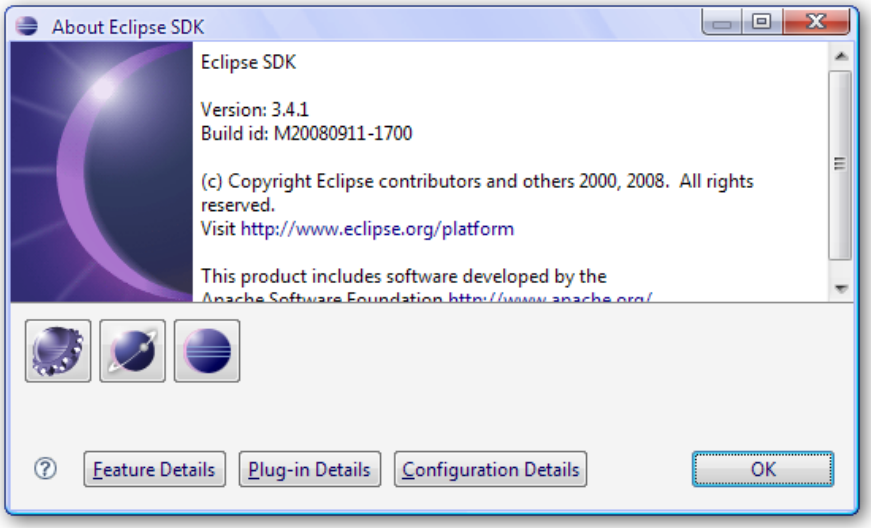


Figura 2: About Eclipse SDK

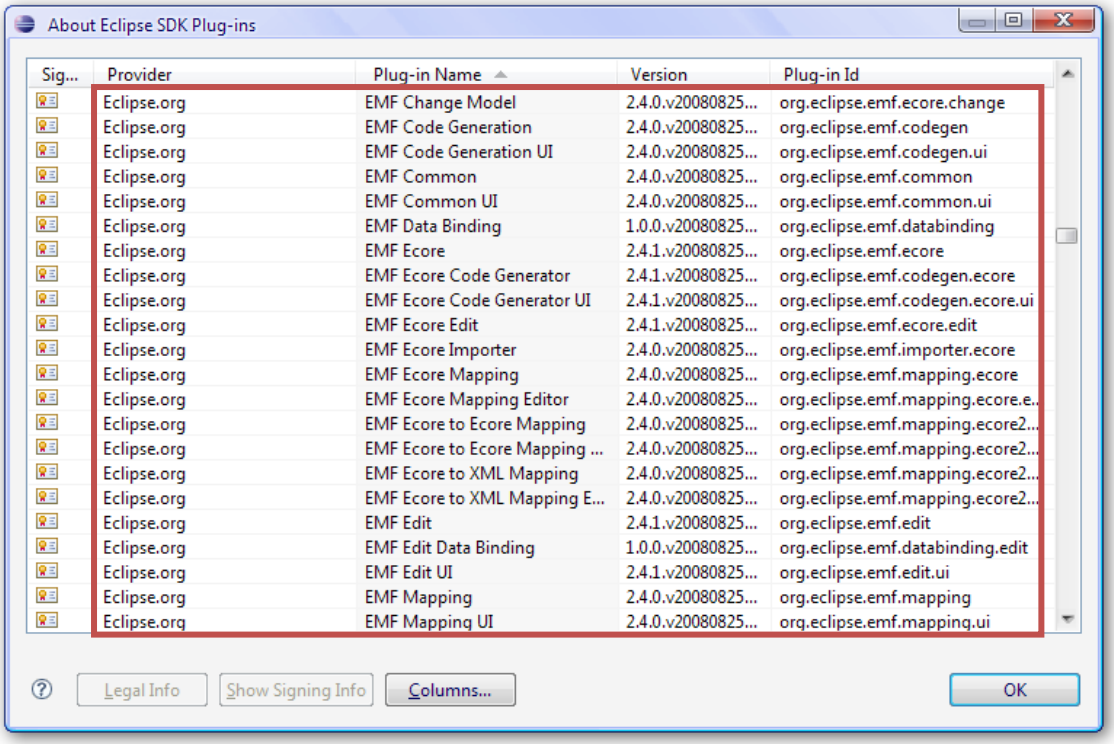


Figura 3: Plug-ins instalados

Ya comprobado que los plug-ins que necesitamos están correctamente instalados, podemos proceder con el tutorial.

3. CASO DE ESTUDIO

A lo largo del tutorial se utiliza un sencillo caso de estudio para ilustrar al lector cada uno de los pasos que ha de realizar. El caso de estudio es muy sencillo para evitar complejidades añadidas, ya que lo que se persigue con este tutorial es aprender el funcionamiento básico de EMF y GMF, y no el desarrollo de un caso de estudio real de forma completa. El caso de estudio consiste en el modelado de la aplicación para un concesionario de coches llamado *InitCar*. Dicho concesionario vende coches, para los que a efectos de este caso de estudio, se considera de interés la marca del coche y el número de matrícula del mismo. Así mismo, es relevante para este tutorial el hecho de que los coches se componen de 4 ruedas y un motor. Las ruedas se caracterizan por el radio que tienen, mientras que el motor se caracteriza por su cilindrada.

4. CREACIÓN DE HERRAMIENTAS ESPECÍFICAS DE DOMINIO

Este tutorial tiene como objetivo enseñar el manejo de EMF y GMF para la creación de herramientas de modelado específicas de dominio. GMF establece un proceso muy específico para construir una herramienta de este tipo. Éste se compone de los siguientes subprocesos: definición del metamodelo o modelo a seguir, generación de código, definición de la metáfora gráfica, definición de las herramientas del modelo, especificación de la correspondencia entre los elementos del modelo y la metáfora gráfica, y generación de la herramienta (ver Figura 4).

Como se puede observar en el diagrama de la Figura 4, el centro del proyecto es el *Domain Model*, es decir el modelo del dominio o metamodelo que será el origen del proceso y del cual se derivarán el resto de subprocesos. El modelo del dominio se define utilizando EMF mediante un lenguaje de definición de modelos denominado Ecore. Todos y cada uno de los subprocesos de GMF están relacionados con el modelo y como indica su nombre, Ecore, constituirá el centro del proyecto en todo momento.

A continuación se explican cada uno de los subprocesos que conforman el desarrollo del proyecto.

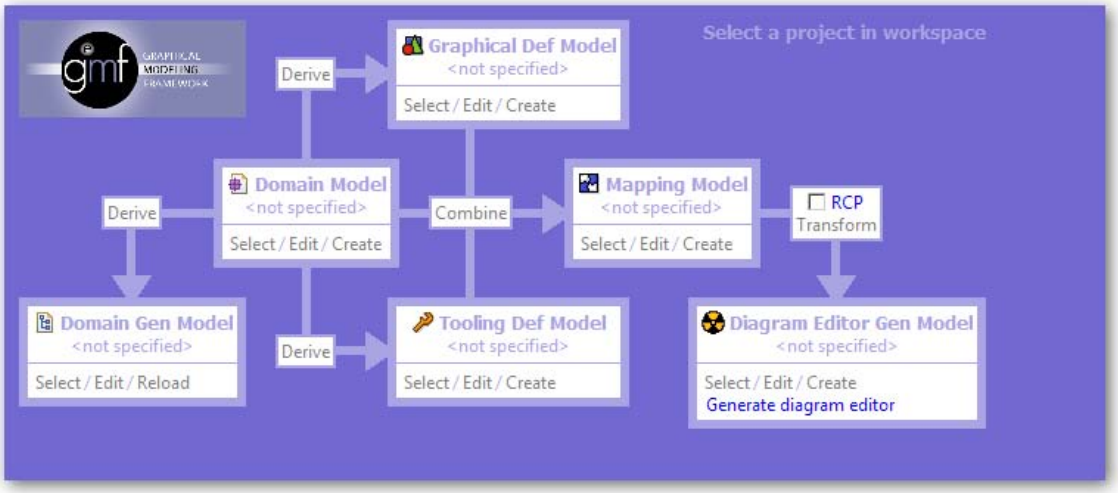


Figura 4: Desarrollo de un proyecto GMF

4.1. Definición del modelo

GMF precisa de un metamodelo y para ello se sirve de EMF, que se trata de un marco de modelado que soporta y genera documentos XMI con un esquema XML Ecore con la especificación del dominio. Es por ello, que este tutorial no sólo es de GMF, sino que también introduce los conocimientos básicos de EMF.

4.2. Definición de la metáfora gráfica

Una vez definido el dominio del modelo (ver Figura 4), se puede empezar a diseñar la definición gráfica de las primitivas de modelado. La definición gráfica consiste en decidir qué primitivas de modelado harán la función de nodos (elementos de la herramienta de modelado que se desea construir), cuáles serán conectores (enlaces entre los elementos de la herramienta de modelado) y cuáles etiquetas (propiedades de los elementos y enlaces de la herramienta de modelado).

4.3. Definición de herramientas

La creación y especificación del panel de herramientas se ha de realizar después de la definición de la metáfora gráfica (ver Figura 4). Consideramos panel de herramientas como la paleta de la herramienta de modelado a crear que proporcionará un funcionalidad “drag and drop”, mostrándonos las primitivas de modelado en la forma que se han diseñado en la definición gráfica. En este paso también se definen los iconos del la herramienta de modelado, que constituyen la iconografía de la paleta.

4.4. Correspondencia entre elementos del modelo y gráficos

La correspondencia entre los elementos del modelo, la metáfora gráfica y herramientas se ha de realizar después de hacer la especificación de las herramientas (ver Figura 4). En este estadio todo lo creado anteriormente cobra un sentido, ya que, que se relacionan y aúnan todos y cada uno de los elementos creados con anterioridad.

Este paso es el último del proceso de creación (ver Figura 4). La creación del generador de la herramienta sin errores indica que todas las anteriores especificaciones se han realizado correctamente y mantienen coherencia.

5. EMF (Eclipse Modeling Framework)

5 Descripción

El Eclipse Modeling Framework (**Framework de Modelado Eclipse, EMF**) es una herramienta que proporciona una estructura de modelado y facilidades para la generación de código al objeto de construir herramientas u otras aplicaciones basadas en un modelo de datos estructurado. A partir de una especificación XML de un modelo, EMF suministra herramientas y soporte de ejecución para producir un conjunto de clases Java en base a ese modelo, un conjunto de clases Adapter, que permiten su visualización y edición basándose en comandos del modelo, y un editor básico.

EMF permite importar modelos que han sido previamente especificados usando documentos Ecore, de los que se hablará más adelante. Una de las características más importantes de EMF es que suministra mecanismos de interoperabilidad con otras herramientas y aplicaciones basadas en EMF.

5.2. Creación de un modelo EMF

Para trabajar con Eclipse siempre es necesario crear un proyecto con el que trabajar, el caso de EMF no es una excepción. Por lo tanto, se ha de crear un proyecto mediante el menú de Eclipse *File →New→Project* (ver Figura 5).

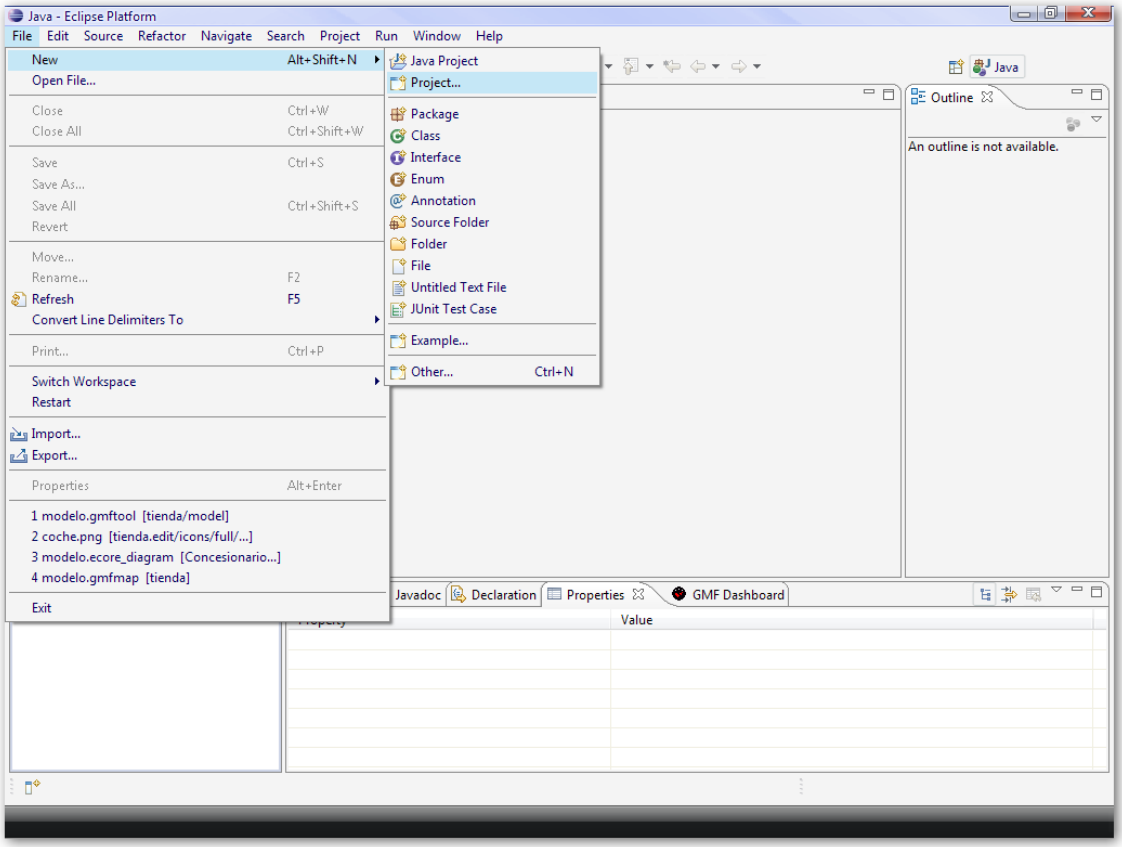


Figura 5: Creación de proyecto en Eclipse

A continuación, se ha de elegir el tipo de proyecto que se desea crear. En nuestro caso se ha de elegir un proyecto EMF vacío, para crear el nuevo proyecto (ver Figura 6). Posteriormente, se pulsa *Next* y se le da nombre al proyecto, acabando la definición del proyecto pulsando el botón *Finish* (ver Figura 7).

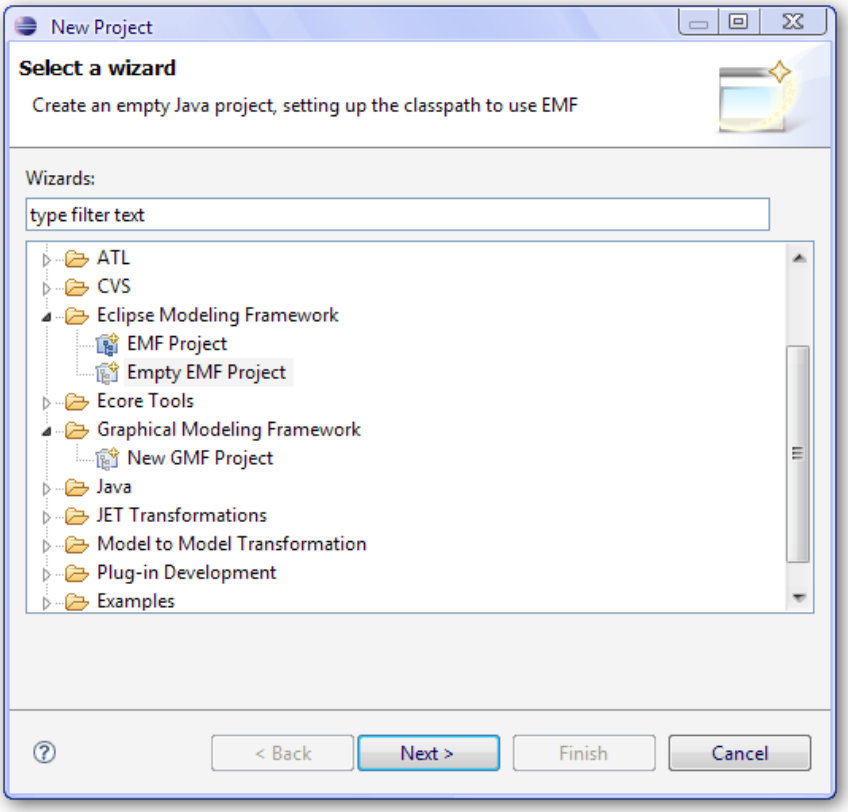


Figura 6: Selección del tipo de proyecto

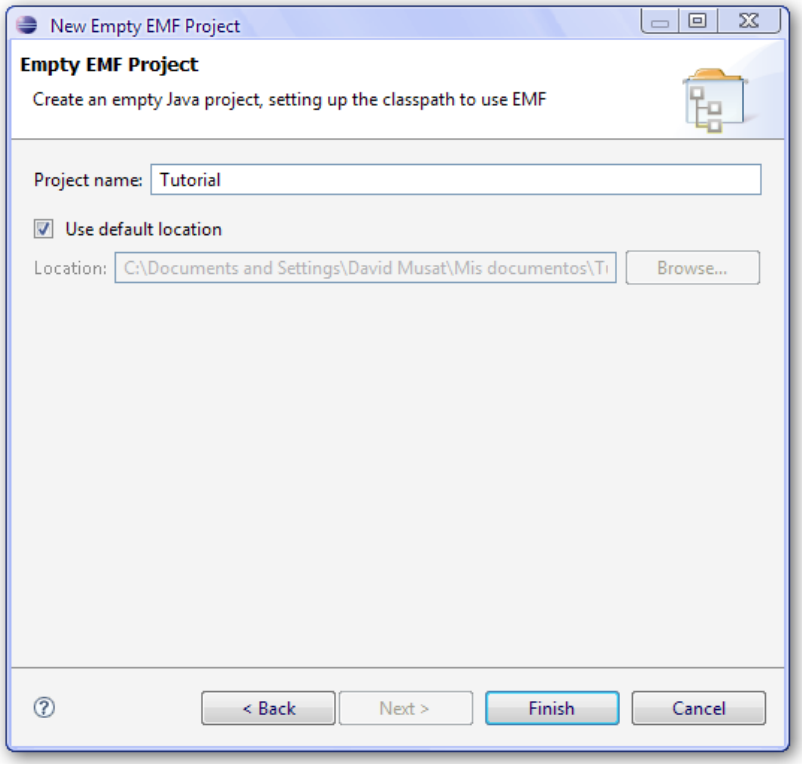


Figura 7: Nombre del proyecto

EMF nos permite crear modelos basados en Ecore[1]. Pulsando con el botón derecho sobre la carpeta model del proyecto creado previamente en *New* → *Other* (ver Figura 8), podemos crear un nuevo modelo seleccionando en la carpeta *Other* el tipo de documento denominado *Ecore Diagram* (ver Figura 9). Este tipo de documento es el que permite especificar modelos Ecore, y para el caso de estudio definir el diagrama Ecore relativo al concesionario. Además, al crear un diagrama Ecore, EMF genera al mismo tiempo un documento XML con extensión .ecore, que asocia al diagrama del modelo, y que es utilizado para la generación de código.

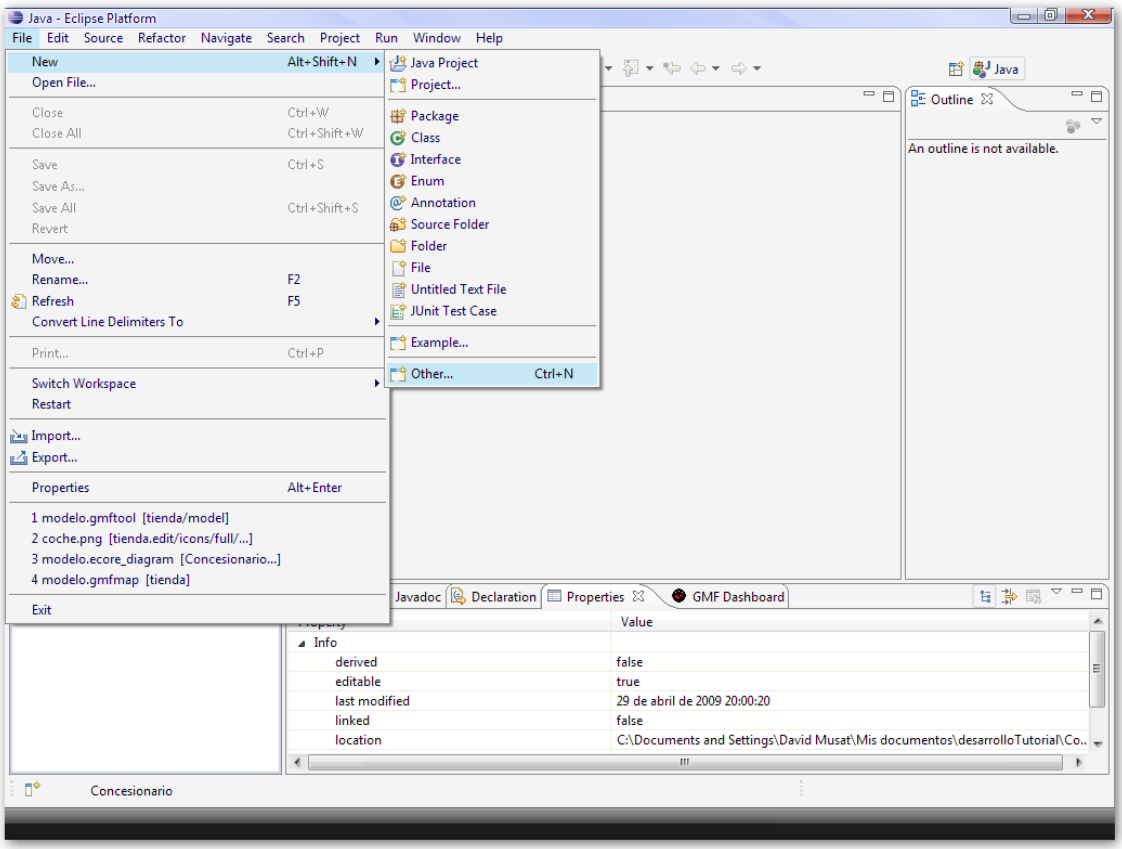


Figura 8: Añadir un elemento al proyecto

Una vez seleccionado el tipo de elemento Ecore Diagram, pulsamos *Next* para posteriormente darle nombre (ver Figura 9). El nombrado consiste en:

- Dar un nombre al documento *.ecore_diagram*, por ejemplo *modelo* y pulsar *Next*.

[1] Metamodelo de EMF para dar soporte al modelado

- Dar el mismo nombre al documento *.ecore* y pulsamos *Next* (podrían tener distintos nombres, pero se realiza así por razones de consistencia).
- Pulsar *Finish*.

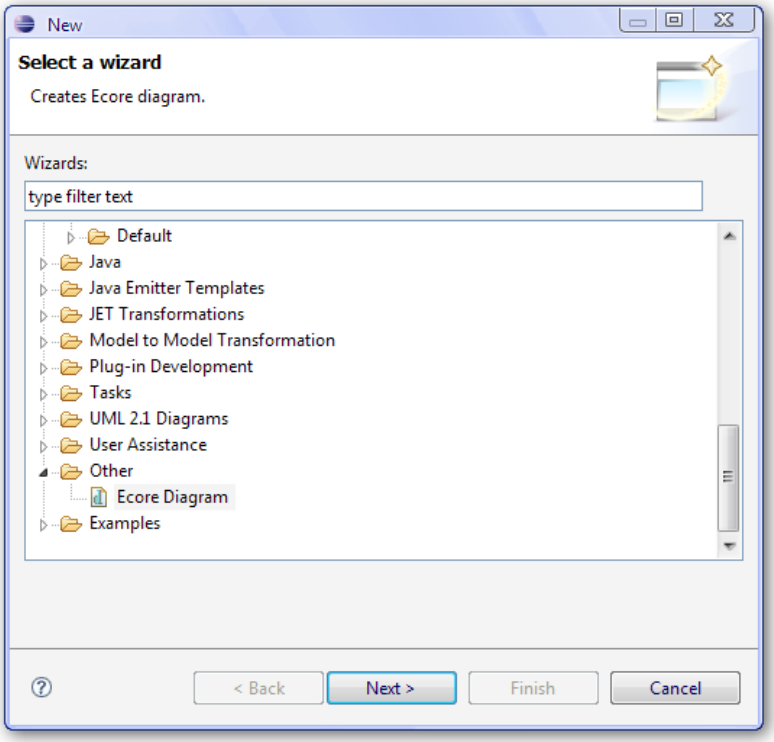


Figura 9: Selección del tipo de elemento que se añade a un proyecto

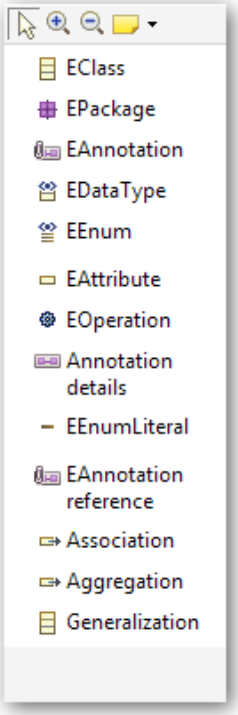


Figura 10: Paleta de herramientas de EMF

Se crean los dos documentos y se abre automáticamente el documento con extensión *.ecore_diagram*. Este documento presenta un tapiz de modelado, y una paleta de herramientas como la de la Figura 10.

La paleta de herramientas incluye los diferentes tipos de elementos y relaciones que permiten definir un modelo Ecore. En la paleta se puede distinguir la herramienta *Eclass*, con la que podremos crear nuevas clases, la herramienta *Epackage*, con la que crearemos nuevos paquetes, *EAnnotation*, con la que se crearán anotaciones y comentarios, *EDataType* con la que crear nuevos tipos de datos, *EAttribute* con la que agregar atributos a las clases, *EOperation* con la que añadir operaciones a las clases y por último, las relaciones de asociación (*Association*), agregación (*Aggregation*) y herencia (*Generalization*).

El modelado se realiza mediante "drag and drop" para cualquiera de las herramientas de la paleta, esto quiere decir que si pulsamos por ejemplo sobre la herramienta de creación de clases y acto seguido pulsamos sobre el tapiz de modelado se nos crea una nueva clase. El nombre de las clases en Eclipse debe comenzar siempre por letra mayúscula. Por otro lado, es posible añadirles tantos atributos y operaciones como se deseen. Hay dos posibilidades de hacerlo: una de ellas consiste en arrastrar desde la paleta de modelado y soltar sobre la clase los atributos y operaciones, y la otra es a través de un menú emergente que aparece superponiendo el puntero sobre la clase en cuestión.

Las relaciones entre clases se instancian gráficamente mediante las 3 últimas herramientas que aparecen en la paleta de modelado. Para poder relacionar dos clases se debe seleccionar la herramienta y acto seguido seleccionar la clase de la que parte la relación (en el caso de herencia, la clase hija, y en el caso de la agregación, la clase compuesta) y después arrastrar y seleccionar la clase destino de la relación (en el caso de herencia, la clase padre, y en el caso de la agregación, la clase componente). Se ha de tener en cuenta, que las asociaciones en EMF son siempre unidireccionales. Si se desea modelar una relación bidireccional, se realiza mediante dos asociaciones que relacionen las dos mismas clases en sentidos opuestos y fusionarlas en una sola

bidireccional. Para ello, se ha de seleccionar dentro de una de la asociaciones, su opuesta en la propiedad *Eopposite*.

Para definir el tipo de un atributo, clase o relación es necesario editar sus propiedades. Esto se realiza pulsando con el botón derecho encima del elemento del que se desean ver las propiedades y seleccionar la opción *Show Properties View*. A continuación, aparece una ventana en la parte inferior del entorno de Eclipse mediante la que se pueden editar todas las propiedades de cada elemento. Entre las propiedades que definen a una clase están el nombre, si pertenecen a alguna subclase, si son de algún tipo predefinido, etc. Para editar el tipo de los atributos, se ha de pinchar en la propiedad *EType* y seleccionar el tipo de datos adecuado para cada atributo, de entre todos los que se muestran en la lista, *string* para las cadenas de caracteres, *int* para los números enteros, *float* para los números reales, *char* para los caracteres y *boolean* para los booleanos, entre otros. Todos los atributos de las clases del caso de estudio son de tipo *String*, excepto el radio de las ruedas que es de tipo *Float* y la cilindrada del motor que es de tipo *Int*.

También, podemos crear enumeraciones de elementos, es decir, listas de tipos de datos. Esta definición se realiza sobre el tapiz de modelado en el que definimos nuestro modelo Ecore. Para ello añadimos un objeto de tipo *EEnum* de la barra de herramientas de modelado al tapiz mediante "drag and drop". Posteriormente, le damos un nombre en el campo *Name*. Este será el nombre que tendrá nuestro tipo de datos enumerado. Para crear la lista sólo nos queda definir dentro del elemento *EEnum* tantos *EEnumLiteral* como número de tipos de datos queramos introducir. *EEnumLiteral* se encuentra también en la barra de herramientas de modelado. A cada *EEnumLiteral* se le ha de asociar su nombre, que será el nombre con el que aparecerá en la enumeración. Para ello, introducimos el mismo nombre para el campo *Name* y *Literal* del *EEnumLiteral*.

Dado que EMF expresa en formato XML el modelo correspondiente a un diagrama ecore, conviene resaltar que para definir un modelo correctamente en EMF es necesario incluir un elemento raíz (una clase) que relacione mediante agregación todas las clases del modelo, en nuestro caso de estudio la clase Concesionario agrega las

clases *Coche*, *Rueda* y *Motor* mediante tres relaciones de agregación. A su vez la clase *Coche*, esta compuesta por la clase *Rueda* y *Motor,* teniendo en cuenta que un coche se compone de cuatro ruedas y un motor. Por ello, al crear una relación se le asigna un nombre y se especifica su cardinalidad. Para esto último, existen dos propiedades dentro de la vista de propiedades, *Lower bound* en la cual se especifica la cardinalidad mínima, y *Upper bound*, en la que se especifica la cardinalidad máxima. Cabe destacar que para conseguir la cardinalidad N, se debe poner el valor -1 y automáticamente, al dejar de editar la relación, se cambia el valor a *.

Una vez conocidas las herramientas de modelado, es posible proceder a modelar el caso de estudio (ver Figura 11).

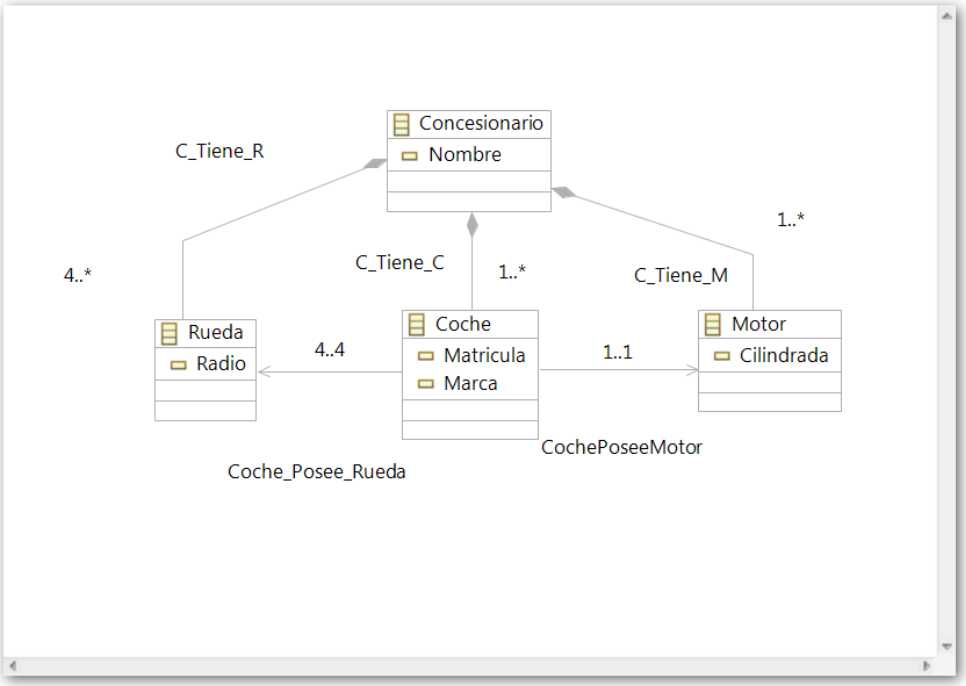


Figura 11: Modelado del caso de estudio.

Por último, hay que dar un identificador al paquete del modelo.ecore. Para ello cerramos el diagrama y hacemos doble clic sobre modelo.ecore en el árbol de documentos de proyecto situado en la ventana izquierda (*Package Explorer*). Si desplegamos totalmente el modelo.ecore podemos ver todas nuestras clases, con sus atributos y relaciones, estructuradas en un árbol (ver Figura 12). Si nos fijamos, hay un

icono con forma de paquete morado en la raíz de árbol con el nombre *null*. Si se hace click con el botón derecho sobre ese paquete y se selecciona *Show Properties View*. Aparece la ventana con sus tres propiedades *Name, Ns Prefix y Ns URI* que se han de rellenar con el nombre del modelo (ver Figura 13).

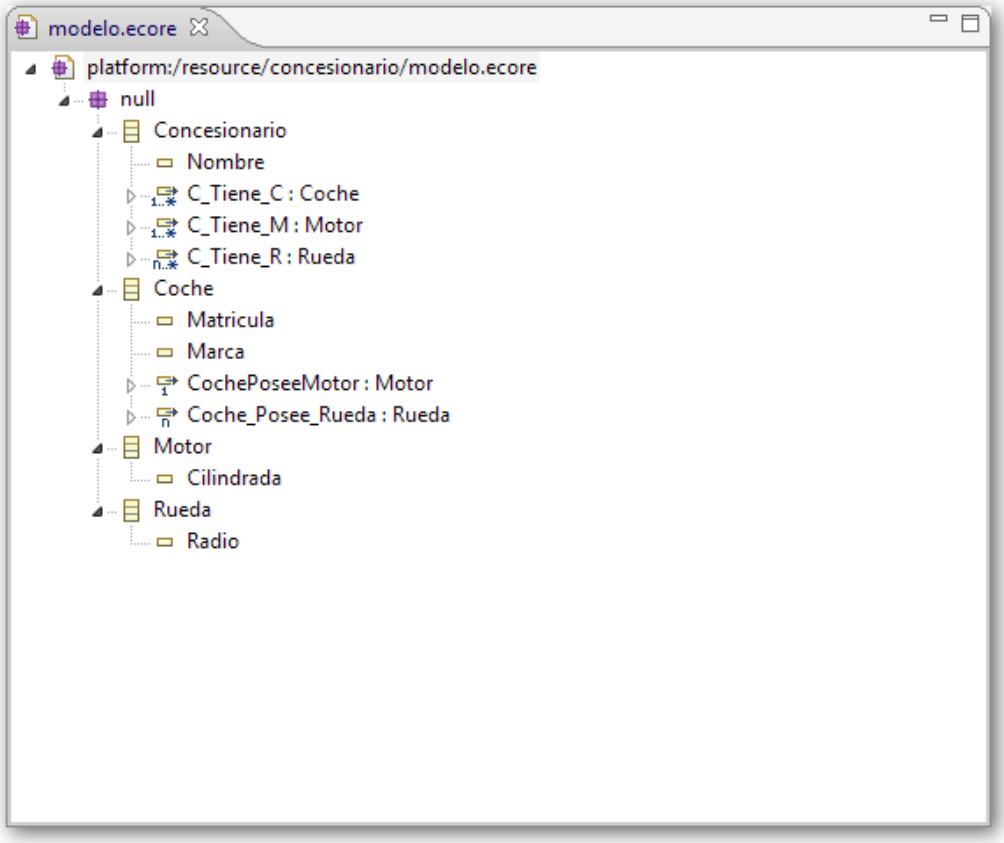


Figura 12: Árbol del documento modelo.ecore

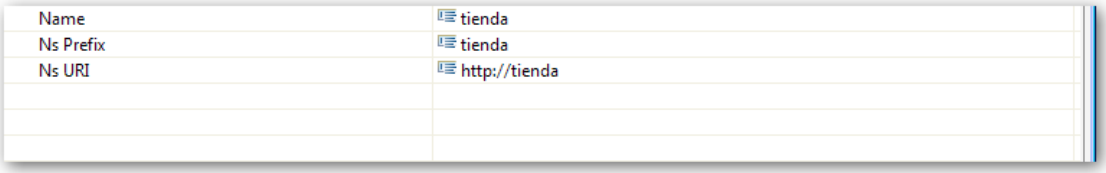


Figura 13: Identificadores del paquete de modelo.ecore

Finalmente, guardamos el proyecto y ya tenemos nuestro modelo EMF creado.

6. GMF (Graphical Modeling Framework)

6.1. Descripción

Graphical Modeling Framework (GMF) proporciona una herramienta para la generación de editores gráficos basados en EMF y GEF. Este último es el que permite al desarrollador crear un editor gráfico completo de forma rápida a partir de un modelo de una aplicación. Una característica destacable en GMF es la reutilización de la definición gráfica para diferentes dominios y aplicaciones, esto es, se pueden reutilizar las metáforas gráficas ya definidas para conceptos de diferentes dominios y aplicaciones. Esta característica se consigue modelando por separado las componentes gráficas que se corresponden con cada uno de los elementos del dominio y la definición de la paleta de herramientas, la cual tendrá una herramienta por cada primitiva. Para completar el proceso de generación de un editor gráfico de dominio, GMF proporciona una definición de mapping o correspondencia mediante la que se asocia cada primitiva de modelado con su componente gráfica y con su herramienta dentro del editor que se está generando.

El proceso de construcción una herramienta de modelado con GMF se ha introducido en la sección 4 A continuación, se procede a aplicar dicho proceso mediante la elaboración de este tutorial.

6.2. Creación de un proyecto GMF

Para crear un nuevo proyecto vamos a *File* → *New* → *Project* y en la carpeta *Graphical Modeling Framework* seleccionamos *New GMF Project*.

- Le damos un nombre al proyecto, en nuestro caso *tienda* y pulsamos *Next*. Es importante que el proyecto no se llame como ninguna de las clases ya que podría dar errores de compilación al generar el código.
- Activamos la casilla *Show dashboard view for the created Project* para obtener una vista del estado de nuestro proyecto como la de la Figura 14. En caso de

que se cierre esta vista, estará accesible desde *Window →Show view → Other→ General → GMF dashboard*.

En este proyecto será donde se cree una herramienta de modelado específico para el caso de estudio utilizado en este tutorial. La pantalla después de los pasos dados debe tener el aspecto mostrado en la Figura 14.

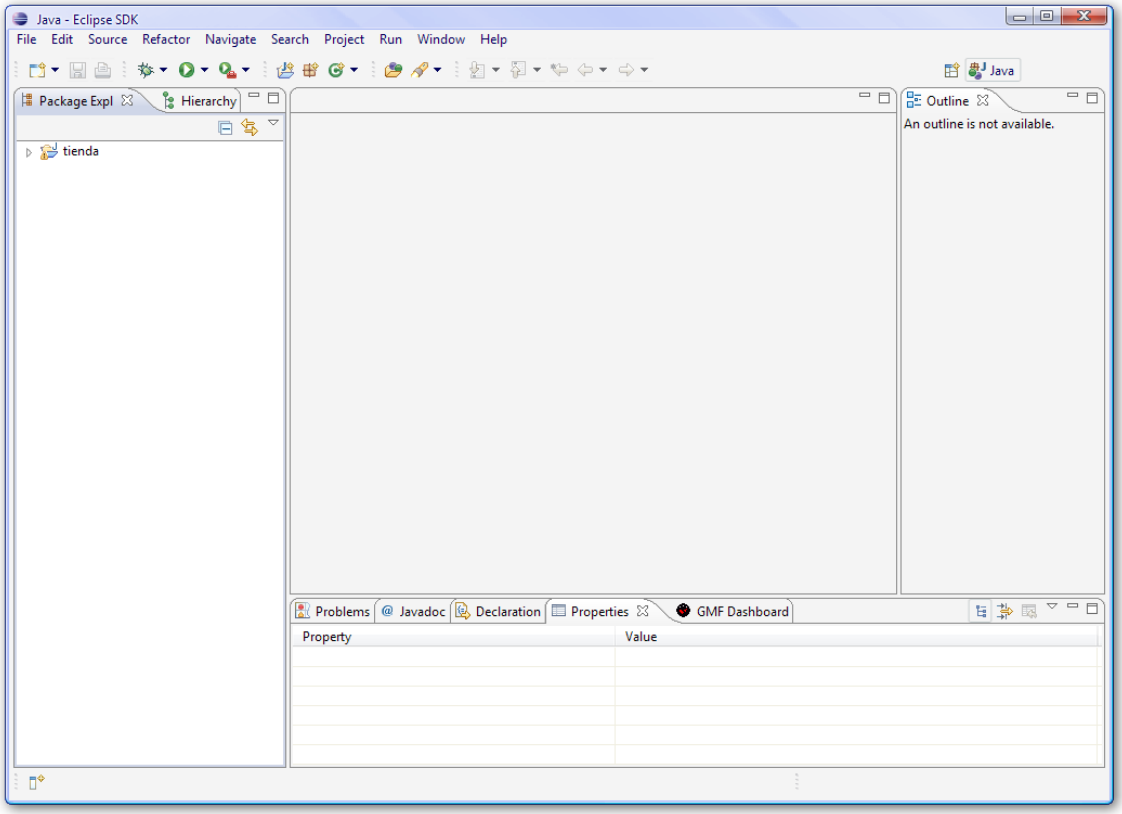


Figura 14: Estado de Eclipse tras la creación de un nuevo proyecto GMF

6.3 Creación o importación de un modelo

El objetivo principal de GMF es proporcionar un editor gráfico para modelar diferentes entornos partiendo de un modelo o metamodelo de entrada. Por esta razón, el centro del proceso de desarrollo de un entorno de modelado en GMF es un modelo. Dicho modelo es posible crearlo dentro de GMF desde cero o bien importar un modelo realizado previamente.

6.3.1. Creación del modelo dentro del proyecto GMF

Para crear el modelo necesitamos una paleta de herramientas que nos permita realizar el diagrama de clases que necesitamos para nuestro proyecto. Si pulsamos con el botón derecho sobre el proyecto, New → Other, encontraremos el documento de tipo Ecore Diagram, pudiendo crear el modelo del mismo modo que se ha presentado en la sección 3.2.

6.3.2. Importación del modelo al proyecto GMF

No es necesario crear un modelo dentro del proyecto raíz GMF, es posible importar un modelo previamente creado. Tanto si se crea desde cero, como se ha explicado en el paso anterior, como si se importa un modelo ya creado, el modelo se ha de definir como el modelo EMF del proyecto GMF (ver sección 5.4). Para ello, Clicamos con el botón derecho sobre el proyecto GMF *New → Other,* en la carpeta Eclipse Modeling Framework seleccionamos *New EMF Model*.

- Seleccionamos *tienda* como directorio raíz, le ponemos como nombre modelo.genmodel y pulsamos *Next*.
- Seleccionamos *Ecore model* como modelo de entrada y pulsamos *Next*.
- Pulsamos *Browse Workspace*, si el modelo Ecore se encuentra en nuestro espacio de trabajo en un proyecto abierto, o *Browse File System* si el modelo está fuera del espacio de trabajo, seleccionamos modelo.ecore como modelo de entrada dentro de la carpeta tienda, y pulsamos *Next*.
- Pulsamos *Finish*.

Si desplegamos el explorador de paquetes, debemos tener una vista similar a la de la Figura 15.

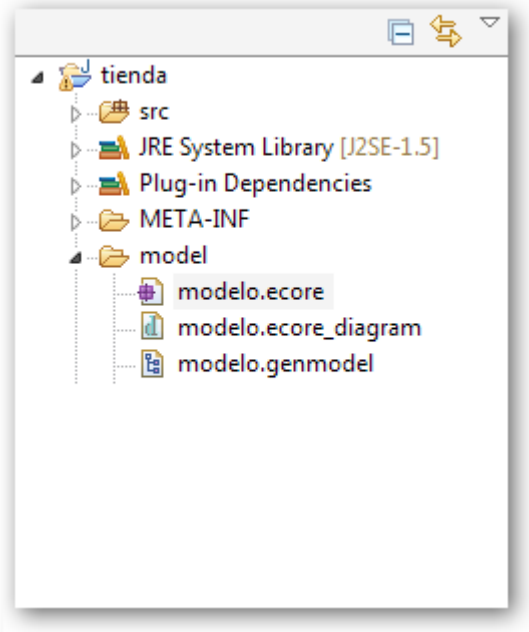


Figura 15: Explorador de paquetes tras la creación del modelo EMF

6.4 Generación del código del modelo

En el punto 6.3.2 se ha creado un nuevo elemento con extensión genmodel. Este elemento es un modelo que permite transformar automáticamente el modelo Ecore que hemos definido a código fuente. El código se genera aplicando patrones de transformación. El resultado de la generación es un conjunto de clases Java, que serán utilizadas más adelante en el proceso de creación de la herramienta de modelado específico de dominio para que todos los elementos que constituyen la herramienta se comporten tal y como el modelo ecore establece. Este proceso de generación se sirve de EMF y es por ello, que en la sección anterior, tanto si el modelo era construido desde cero como si no, el modelo se ha importado como un modelo EMF dentro del proyecto.

Antes de generar el código del modelo es preciso configurar el modelo genmodel para especificar su paquete base. Para ello hacemos doble clic en *modelo.genmodel*. Se abre una ventana con un árbol similar al de modelo.ecore. Clicamos con el botón derecho sobre el paquete morado, de nombre *Tienda*, y seleccionamos *Show properties view*. Se nos abre una tabla que debemos rellenar como la Figura 16.

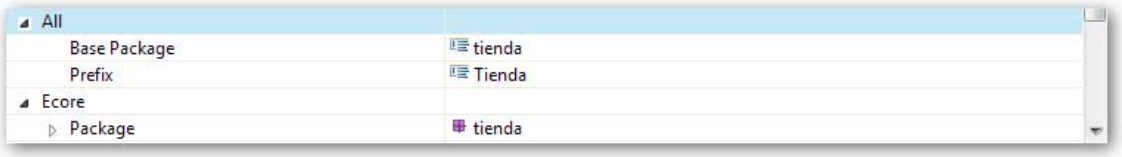


Figura 16: Tabla de propiedades del modelo EMF

Ahora está todo listo para generar el código del modelo. Clicamos con el botón derecho sobre el paquete morado y seleccionamos *Generate Model Code*. Se generará automáticamente un plug-in con el código del modelo. A continuación, realizamos el mismo proceso seleccionando *Generate Edit Code* y *Generate Editor Code*. Este último será el intérprete de Eclipse que nos permita ver nuestro modelo específico de dominio en forma de árbol tal y como nos muestra los documentos Ecore y cualquier componente de GFM.

6.5 Definición gráfica del modelo

La definición gráfica del modelo consiste en definir el aspecto que van a tener las primitivas de modelado en nuestro editor gráfico. Así, GMF de una forma totalmente automática, nos permite personalizar el aspecto de la aplicación de construcción de modelos específicos de dominio a nuestro gusto.

Para este tutorial se han elegido las siguientes metáforas gráficas:

- Primitiva gráfica *Coche*: Rectángulo con picos romos, bordes de color azul oscuro y lineal e interior de color azul claro.
- Primitiva gráfica *Rueda*: Elipse con los bordes punteados de color gris oscuro e interior de color gris claro.
- Primitiva gráfica *Motor*: Rectángulo, con los bordes de rayas discontinuas color naranja e interior amarillo claro.
- Primitiva gráfica *Coche_Posee_Rueda*: Línea fina de color azul claro.
- Primitiva gráfica *CochePoseeMotor*: Línea gruesa discontinua de color azul oscuro.

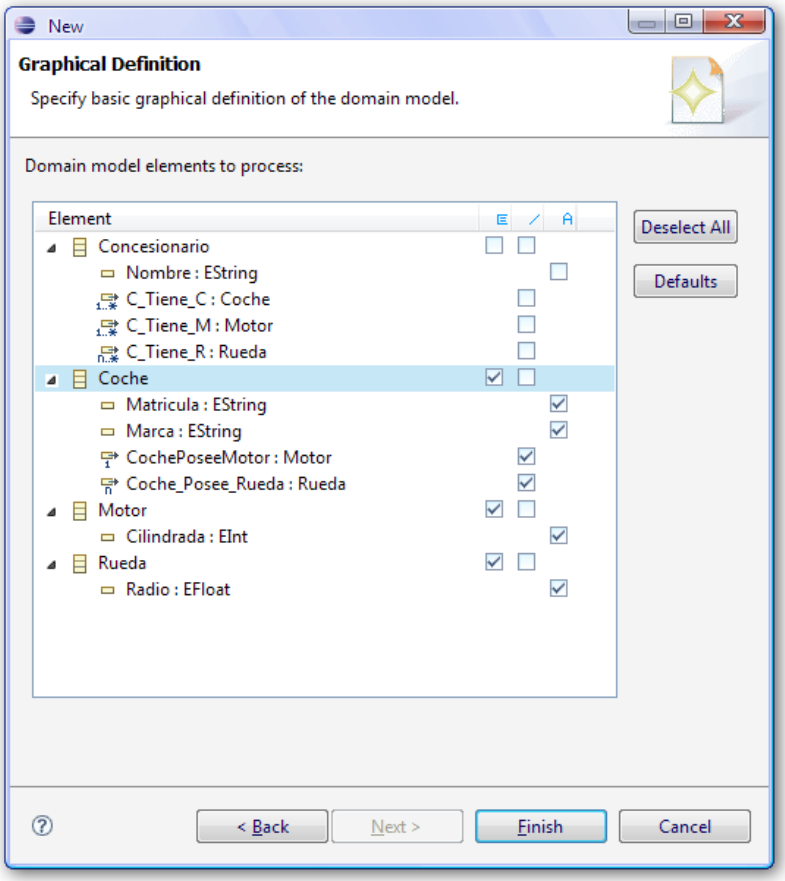


Figura 17: Selección del comportamiento de las primitivas

Una vez elegido el aspecto de la aplicación, podemos empezar a crear la definición gráfica del modelo. Para ello utilizamos *File* → *New* → *Other*, vamos a la carpeta *Graphical Modeling Framework* y seleccionamos *Simple Graphical Definition Model* y pulsamos *Next*.

- Cambiamos el nombre a *modelo.gmfgraph* y pulsamos *Next* comprobando que el directorio raíz marcado es *tienda*.
- Seleccionamos *modelo.ecore* como modelo de entrada, si no aparece por defecto, pulsamos en *Find in Workspace* para seleccionarlo. Después, seleccionamos el elemento raíz de nuestro modelo, es decir, el que contiene al resto, en nuestro caso es *Concesionario*, pulsamos *Next*.
- En este paso nos aparecerá una ventana como la de la Figura 17, en la que podremos elegir qué elementos del modelo del dominio actuarán como nodos, cuáles como enlaces y cuáles como etiquetas. En la ventana aparecen representados en una tabla de tres columnas en la que se encuentran

colocados de izquierda a derecha, y están representados gráficamente en la cabecera de la siguiente forma: cuadrado con rayas horizontales (nodos), línea inclinada (enlaces), y letra A (etiquetas). Cada elemento del modelo tiene asociada una *checkbox* (casilla de selección) en cada columna de la tabla que es posible seleccionar, para así elegir el tipo de representación gráfica que se desea. En nuestro caso, hacemos la selección de la Figura 17 y pulsamos *Finish*.

Al pulsar *Finish* aparecerá en la ventana el *modelo.gmfgraph*. Este tiene estructura de árbol, si desplegamos el elemento raíz, nos aparecerá una vista similar a la de la Figura 18.

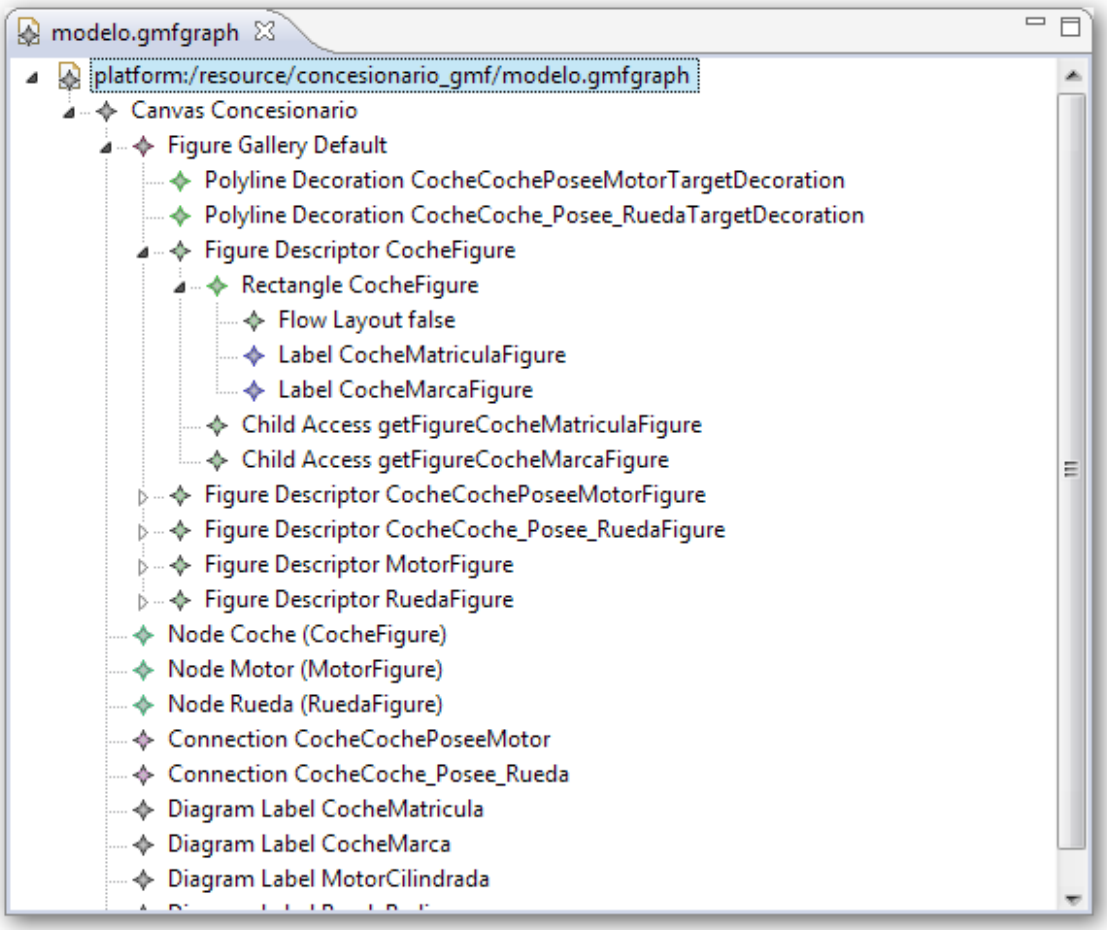


Figura 18: Árbol de definición gráfica

A continuación, seleccionamos *Figure Gallery Default* observando que los enlaces van determinados por un *Polyline Decoration* y el nombre de la primitiva que representa ese enlace. Desplegando los nodos del modelo (*Coche, Rueda, Motor*) vemos que

vienen determinados por un descriptor de figura que contiene la figura *Rectangle* y contiene las funciones de acceso a esas etiquetas *Child Access getFigure*. Si a su vez la desplegamos la figura *Rectangle* de cada nodo, vemos que contiene sus etiquetas *Label* y la alineación de las etiquetas *Flow Layout*, A parte de esto, por cada figura existe un nodo *Node* de acceso al *FigureDescriptor* y por cada enlace una *Connection* de acceso al *Polyline Decoration*.

Sabiendo todo esto, vamos a personalizar la interfaz de la aplicación. Para empezar, vamos a crear el diseño del coche.

- Clicamos con el botón derecho sobre *Figure Gallery Default*.
- Seleccionamos *New Child*.
- Seleccionamos *Figure Descriptor*.
- Para personalizar las metáforas gráficas de las primitivas de modelado crearemos nuevos descriptores gráficos. Por ejemplo, para la primitiva Coche, se crea un nuevo descriptor, pinchamos con el botón derecho sobre él y seleccionamos Show Properties View. Le ponemos el nombre del descriptor que tenemos del coche, *CocheFigure*.
- Clicamos sobre el descriptor de nuevo con el botón derecho y seleccionamos *New Child* → *Rounded Rectangle*, ya que era lo que habíamos definido en el diseño y le pondremos *Coche* de nombre.
- Seleccionamos todos los elementos del *FigureDescriptor CocheFigure* anterior y los trasladamos arrastrándolos al lugar donde corresponden (el nuevo descriptor creado), excepto el *Rectangle CocheFigure*. Por lo tanto, las *Labels* y el *Flow Layout* irán en la jerarquía dentro del *Rounded Rectangle,* y los *Child Access* irán dentro del *Figure Descriptor.*
- Una vez que hemos trasladado todos los elementos al nuevo Descriptor, borramos el antiguo (ver Figura 19).

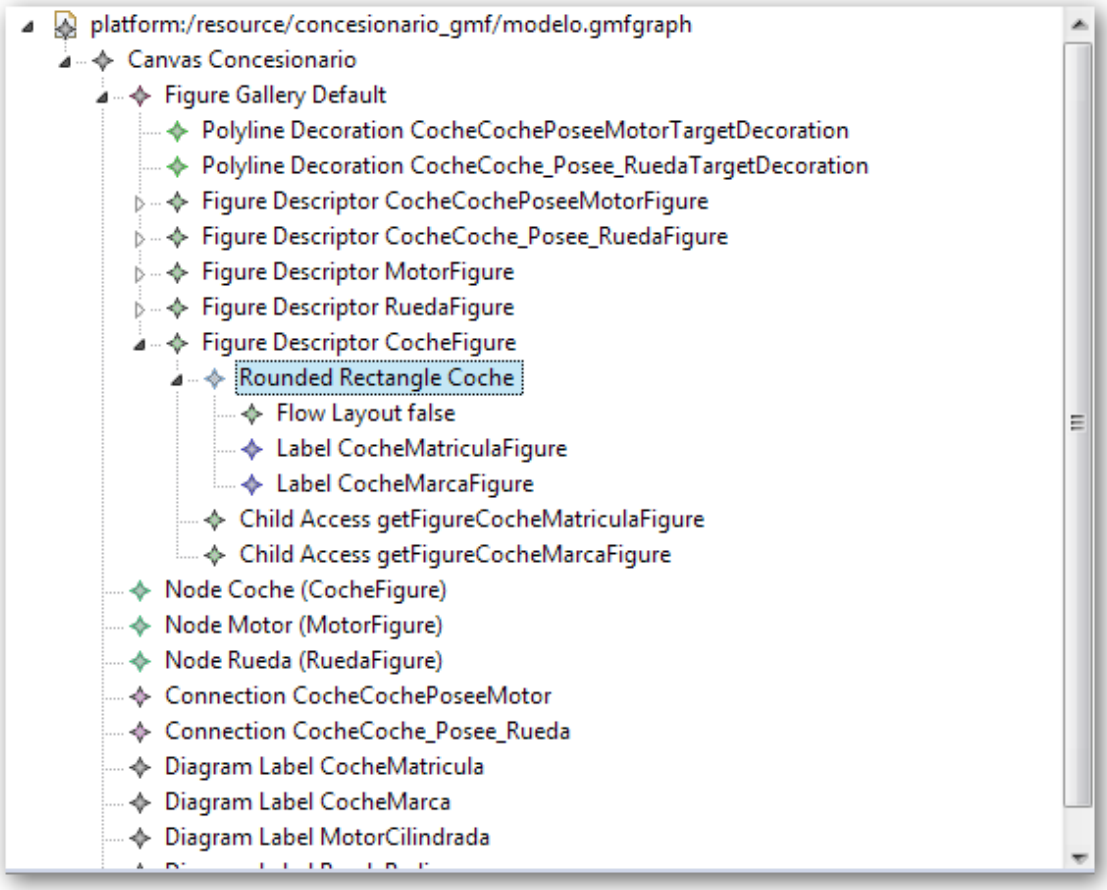


Figura 19: Creación de un nuevo descriptor

Ahora sólo queda darle color. Para ello, pulsamos con el botón derecho sobre el *Rounded Rectangle Coche* → *New Child* → *Foreground RGB color* e introducimos los valores de la Figura 20. Estos valores se pueden consultar por ejemplo con una herramienta de dibujo como por ejemplo PaintBrush de Windows.

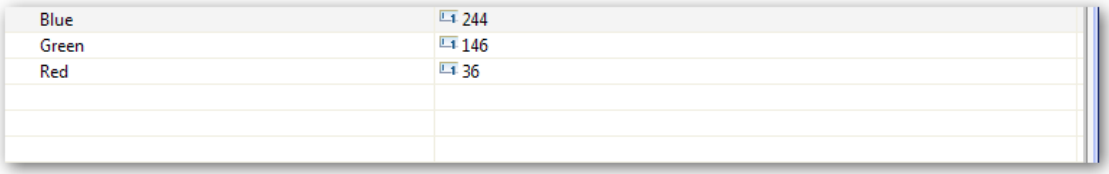


Figura 20: Propiedades de Foreground RGB Color

Seguimos el mismo proceso seleccionando *Background RGB Color*. Introducimos los valores de la Figura 21.

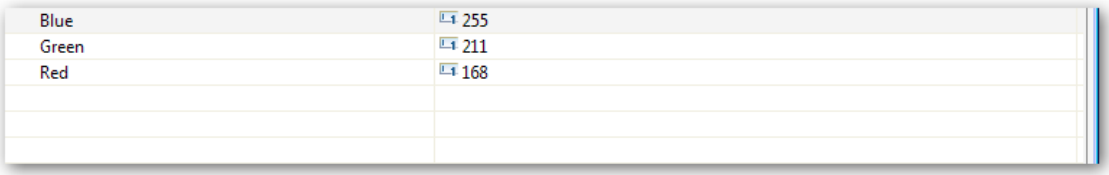
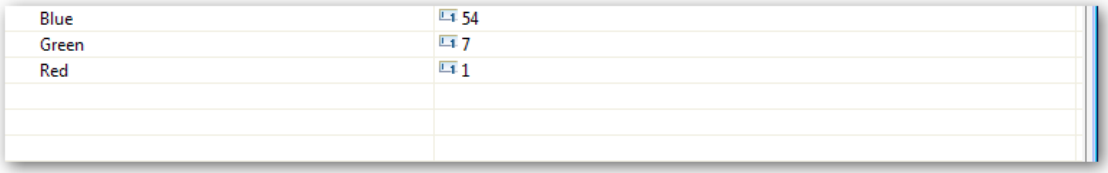


Figura 21: Propiedades de Background RGB Color

Por último borramos *Flow layout* e insertamos *Grid layout* de la misma forma, a través de *New Child*. Esto se hace para que las etiquetas estén en columnas y no en filas (ver resultado en Figura 25). Con el resto de nodos, seguimos el mismo proceso para definir su primitiva gráfica.

Ahora haremos la definición gráfica de un enlace, en particular de *CochePoseeMotor*. Para ello desplegamos el descriptor de ese enlace y clicamos sobre *PolylineConnection* añadiendo un *Background RGB color* con los valores de la Figura 22.

Figura 22: Valores para el Background de Coche_Posee_Motor



Y por último, ya que este enlace tiene un grosor determinado y es punteado, lo modificaremos clicando en *Polyline Connection* con el botón derecho y *Show Porperties View*. Pondremos los valores de la Figura 23.

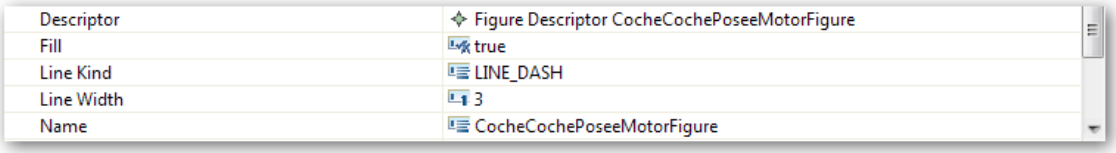


Figura 23: Valores para la definición de una figura

La definición de los bordes del resto de las figuras se realiza igual, accediendo a las propiedades y cambiando el *Line Kind* y el *Line Width*.

Una vez realizado el diseño hay que comprobar que los nodos de acceso a las figuras apuntan a las figuras correctas. Para ello, vamos a *Node Coche*, hacemos clic con el botón derecho y seleccionamos *Show properties view*. En el campo *Figure*, desplegamos y seleccionamos la figura *CocheFigure* (ver Figura 24). También se ha de comprobar que los *Diagram Label* apuntan a la etiqueta y a la figura (*Figure*) a la que representan.

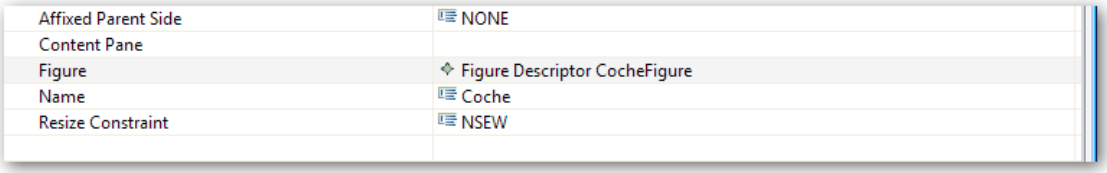


Figura 24: Definición de los nodos

Tras definir los nodos ya tendremos la definición gráfica completa. En la Figura 25 se muestra la vista del árbol desplegado.

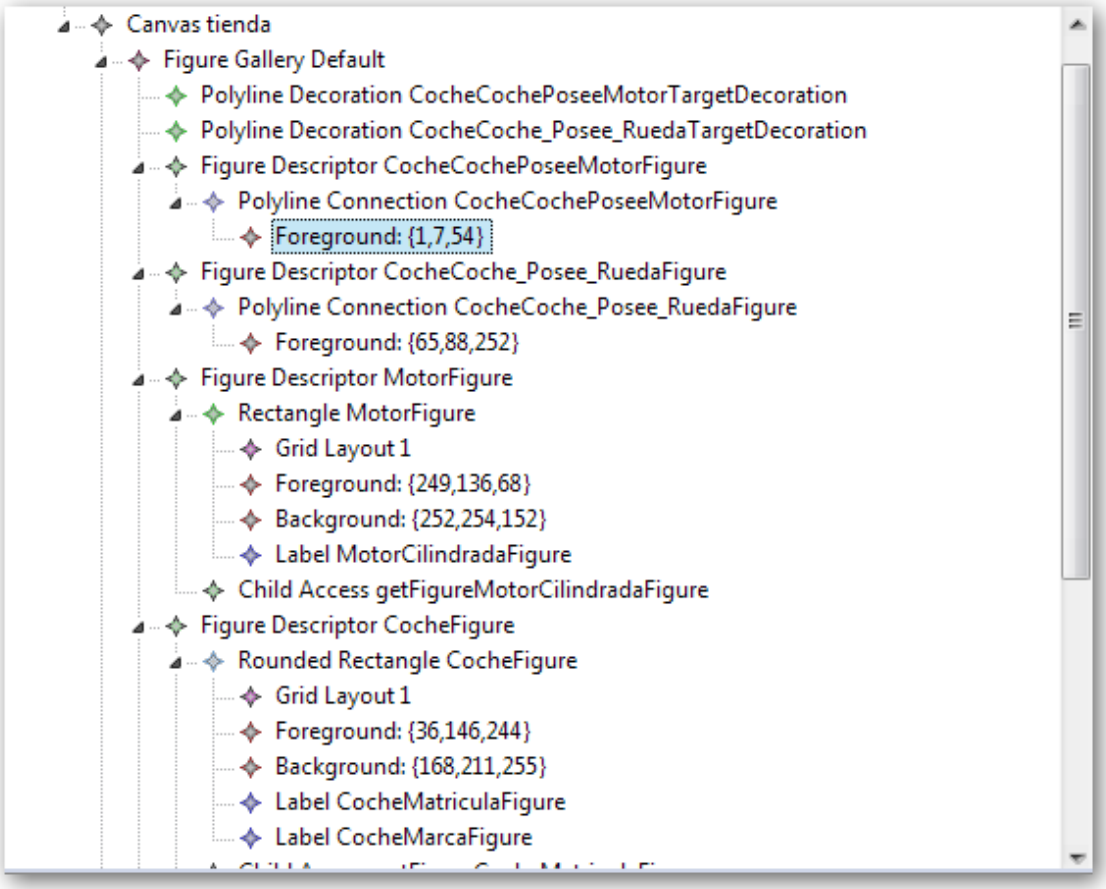


Figura 25: Árbol de definición gráfica desplegado

Antes de pasar al siguiente apartado es conveniente validar la definición gráfica anterior. Para ello, se ha de pinchar sobre *Canvas Concesionario* con el botón derecho y seleccionar *Validate*. Si es correcto, pasar al siguiente punto, si no comprobar los campos *Figure* de los nodos y los *Diagram Label*.

6.6 Definición de la paleta de herramientas de modelado

En este apartado definiremos la paleta de la aplicación con la que construiremos los modelos. Para ello, se ha de seleccionar *File* → *New* → *Other,* nos dirigimos a la carpeta Graphical Modeling Framework y seleccionamos *Simple Tooling Definition Model* y pulsamos *Next*.

- Llamamos a la definición de la paleta de herramientas *modelo.gmftool* y pulsamos *Next*, comprobando que el directorio raíz marcado es *tienda*.
- Seleccionamos *modelo.ecore* como modelo de entrada, si no aparece por defecto, pulsamos en *Find in Workspace* para seleccionarlo. Después, seleccionamos el elemento raíz de nuestro modelo, es decir, el que contiene al resto, en nuestro caso es *Concesionario*, pulsamos *Next*.
- Al igual que en la definición gráfica, se nos presentan los elementos del modelo y nos permite elegir aquellos que van aparecer como nodos o como enlaces en la barra de herramientas. En nuestro caso, marcamos la selección como en la Figura 26 y pulsamos *Finish*.

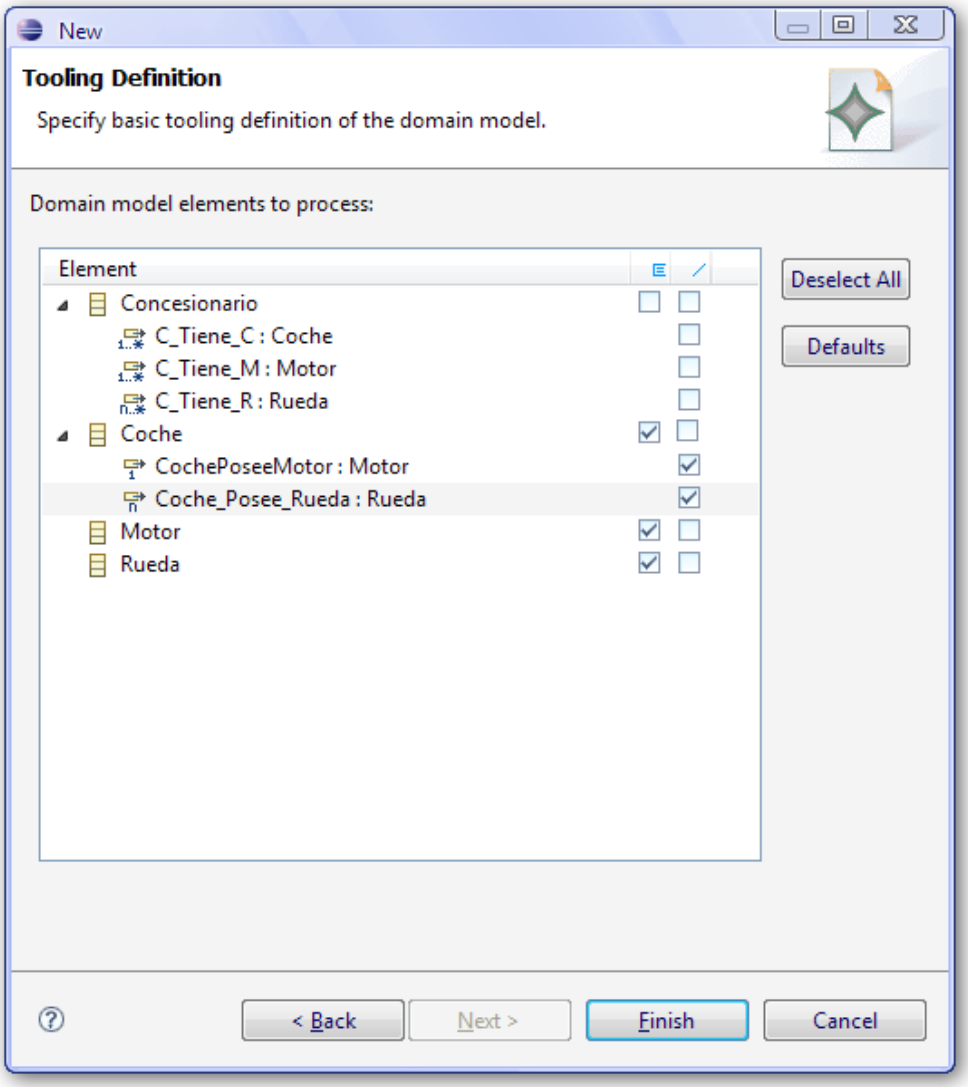


Figura 26: Definición del comportamiento de las primitivas en la paleta de herramientas

Nos aparece un nuevo árbol, que representa la definición de herramientas de la aplicación a crear (ver Figura 27).

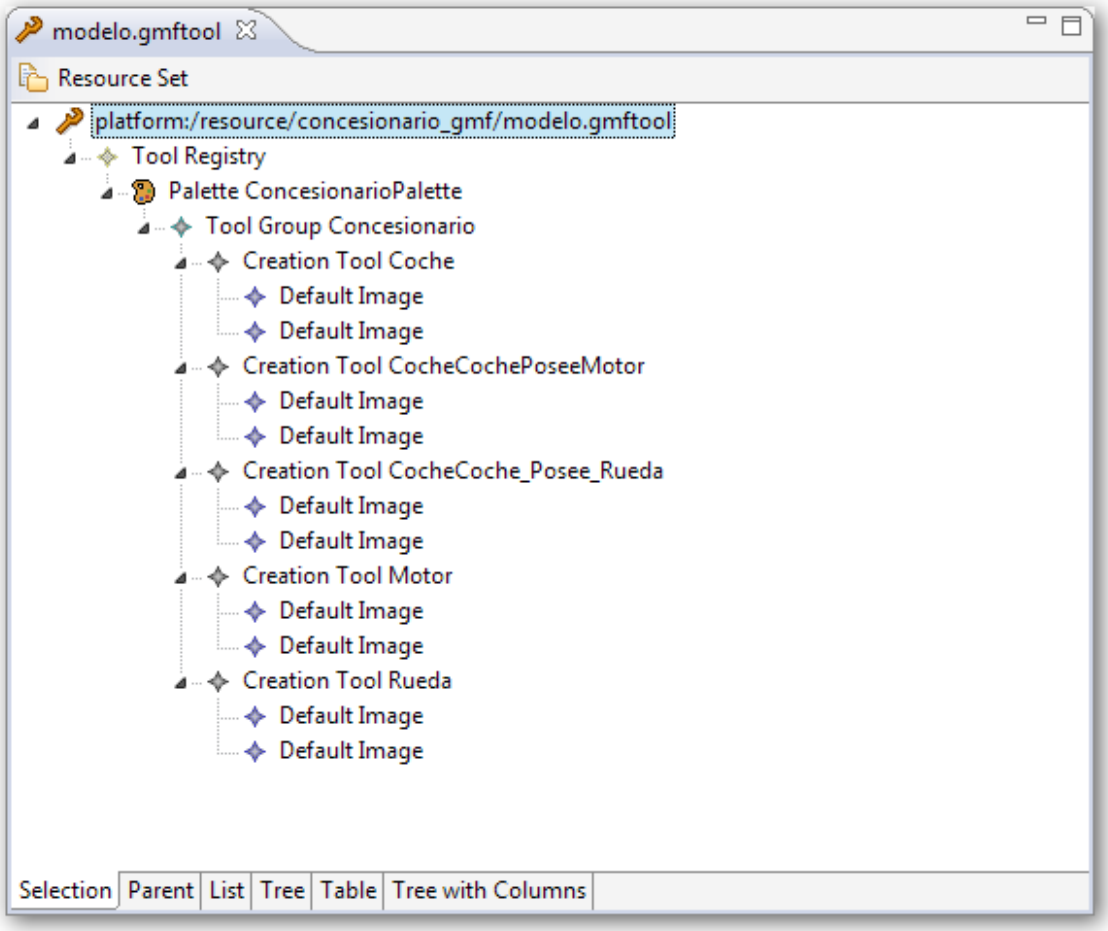


Figura 27: Árbol de definición de herramientas

Cabe destacar que por cada primitiva seleccionada se crea una herramienta de creación y dos imágenes por defecto que corresponden a los iconos de la aplicación. El icono por defecto es ✦.

Para distinguir unas herramientas de otras vamos a asignarles distintos iconos. Para ello, se han de copiar todos los iconos que se distribuyen junto a este tutorial a la carpeta tienda.edit/icons/full/obj16.

Posteriormente, procedemos a asignarle a cada nodo su icono. A continuación, mostramos un ejemplo para el caso del nodo *Coche*:

- Borramos los dos *default image* que cuelgan de *Creation Tool Coche*.

- Clicamos con el botón derecho sobre *Creation Tool Coche*, seleccionamos *New Child* y creamos un *Small Icon Bundle Image* y un *Large Icon Bundle Image*.
- Definimos las propiedades de ambos como muestra la Figura 28.

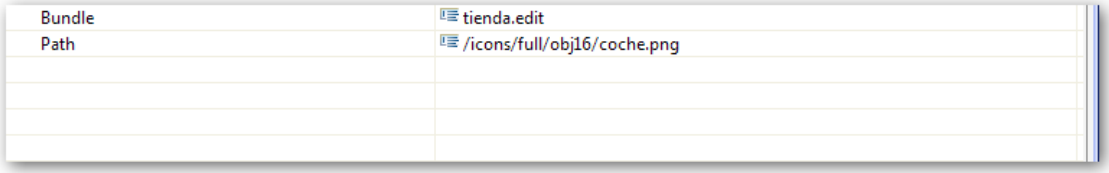


Figura 28: Propiedades de Large y Low Icon Bundle Image

El resto de iconos tienen la misma ruta y hay que cambiar el nombre según el nombre del archivo.

Una vez hecho todo el proceso con el resto de las primitivas, el árbol de definición debe quedar como el que aparece en la Figura 29. A continuación se ha de validar la definición de la barra de herramientas. Para ello, se ha de pinchar sobre *Tool Registry* con el botón derecho y seleccionar *Validate*.

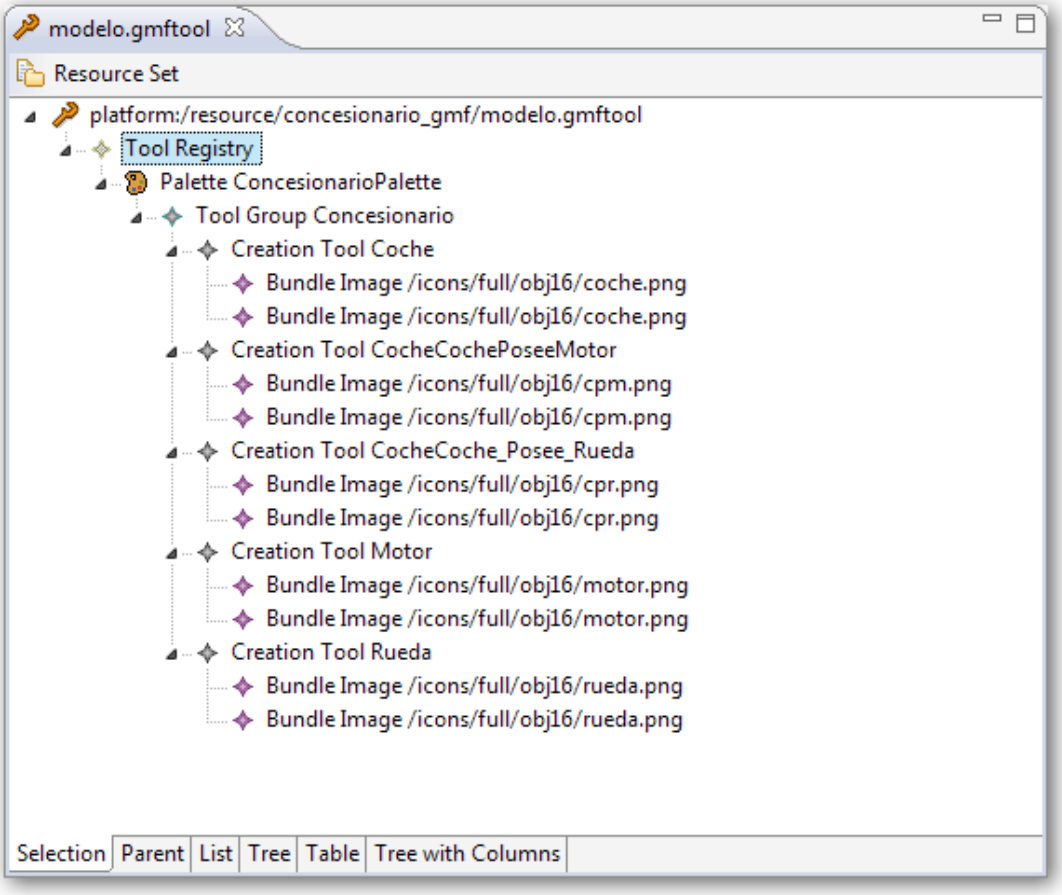


Figura 29: Árbol de definición de herramientas

6.7 Mapping

El proceso de mapping o correspondencia es aquel en el que todo lo que hemos creado cobra un sentido y se une para formar la herramienta de construcción de modelos específicos de dominio. Será en este paso donde conozcamos si toda la creación de esta herramienta tiene consistencia.

Para comenzar, se ha de clickar en *File →New → Other* y en la carpeta *Graphical Modeling Framework* seleccionar *Guide Mapping Model Creation*.

- Seleccionamos *tienda* como carpeta raíz y llamamos al documento *modelo.gmfmap*. Pulsamos *Next*.
- Seleccionamos *modelo.ecore* como modelo de entrada, si no aparece por defecto, pulsamos en *Find in Workspace* para seleccionarlo. Después, seleccionamos el elemento raíz de nuestro modelo, es decir, el que contiene al resto, en nuestro caso es *Concesionario*, pulsamos *Next*.
- Seleccionamos el *modelo.gmftool,* si no aparece por defecto, pulsamos en *Find in Workspace* para seleccionarlo, y pulsamos *Next*.
- Seleccionamos modelo.gmfgraph, si no aparece por defecto, pulsamos en *Find in Workspace* para seleccionarlo, y pulsamos *Next*.
- A continuación aparece una ventana como la de la Figura 30, en la que se especifican los nodos y los enlaces *(links).*
- Marcamos *Coche* y pulsamos *Change*.
- Cambiamos los valores de *Diagram Node* y *Tool* a las metáforas y herramientas que hemos definido para *Coche* y hacemos lo mismo con *Rueda y Motor*. Pulsamos *Finish*.

Una vez que hayamos pulsado *Finish* nos aparecerá el árbol de especificación del proceso de mapping (Figura 31). En este paso debemos comprobar que el mapping se ha realizado correctamente. Para ello, se ha de pinchar sobre *Mapping* con el botón derecho y seleccionar *Validate*.

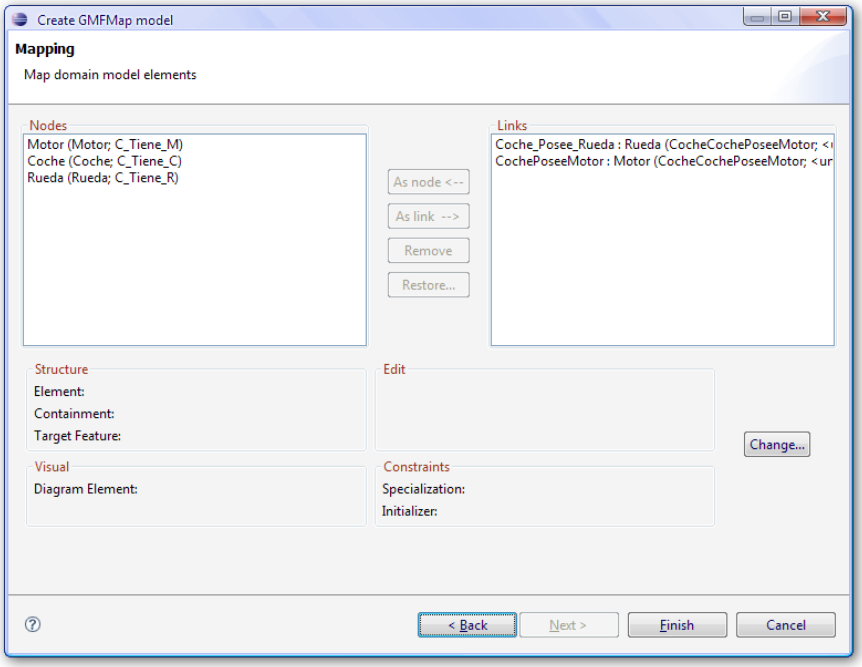


Figura 30: Diseño de los nodos y los enlaces en el proceso de mapping

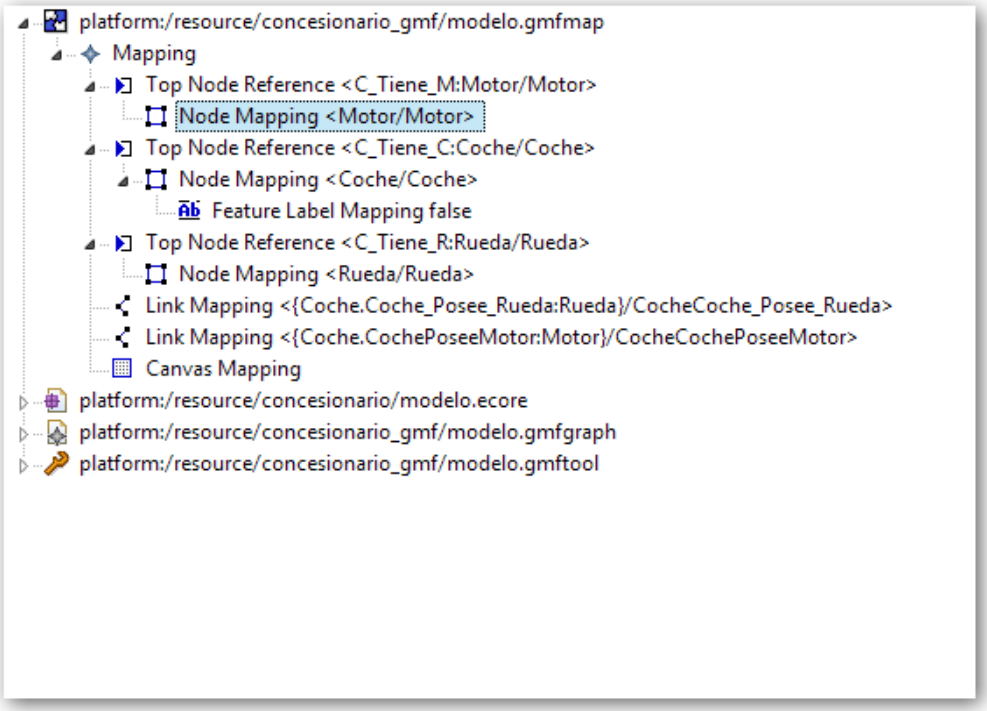


Figura 31: Árbol de especificación del proceso de mapping.

Como se puede observar en la Figura 31, cada nodo cuelga de un nodo de referencia que indica qué relación existe entre el *Concesionario* y el nodo en cuestión. Si nos fijamos un poco más en detalle, podemos darnos cuenta que muchas de las etiquetas

que hemos declarado en el modelo no aparecen. Este es el momento de crearlas. Vamos a crear la etiqueta de matrícula o marca del coche, ya que si pinchamos sobre el *Feature Label Mapping false* que aparece anidada dentro de *Node mapping <Coche/Coche>* veremos que una de ellas, matrícula o marca, está creada. Por lo tanto, hay que añadir la que nos falte, en este caso vamos a suponer que sea *matricula*:

- Pinchamos con el botón derecho sobre *Node mapping <Coche/Coche>*.
- Seleccionamos *New Child* → *Feature label mapping*.
- Vamos a propiedades. En el campo *Features* seleccionamos *matrícula* y pulsamos *add*. A continuación pulsamos *OK*. Cambiamos el campo *Diagram label* por la etiqueta que corresponda, en este caso *Diagram Label CocheMatricula*.

El proceso es similar para el resto de los nodos, ya que en el resto no aparecen las etiquetas, y hay que añadirlas

A continuación hay que comprobar que cada nodo está asociado con su herramienta de creación y su componente gráfica. Veamos el caso del *Coche*.

- Pulsamos con el botón derecho sobre el *Link mapping <{Coche.CochePoseeMotor:Motor}/CocheCochePoseeMotor>*.
- Seleccionamos *Show Properties View*.
- Observamos si los campos *Diagram Link* y *Tool* coinciden con el enlace.
- Si no es así lo cambiamos. En el caso de CochePoseeMotor quedaría como en la Figura 32.

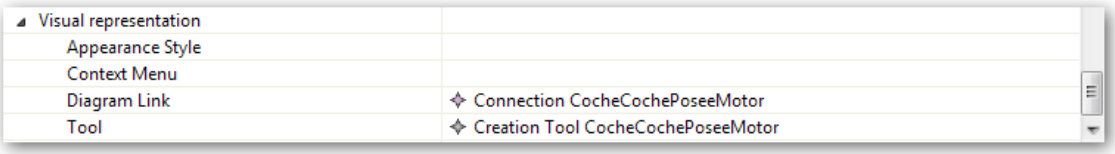


Figura 32: Proceso de mapping del nodo Coche.

Este proceso se debe realizar con todos los nodos y los enlaces. Una vez completado se ha de proceder a la validación del modelo de correspondencias. Validamos el proceso del mismo modo que en el punto 3. El árbol desplegado debe quedar como el de la Figura 33.

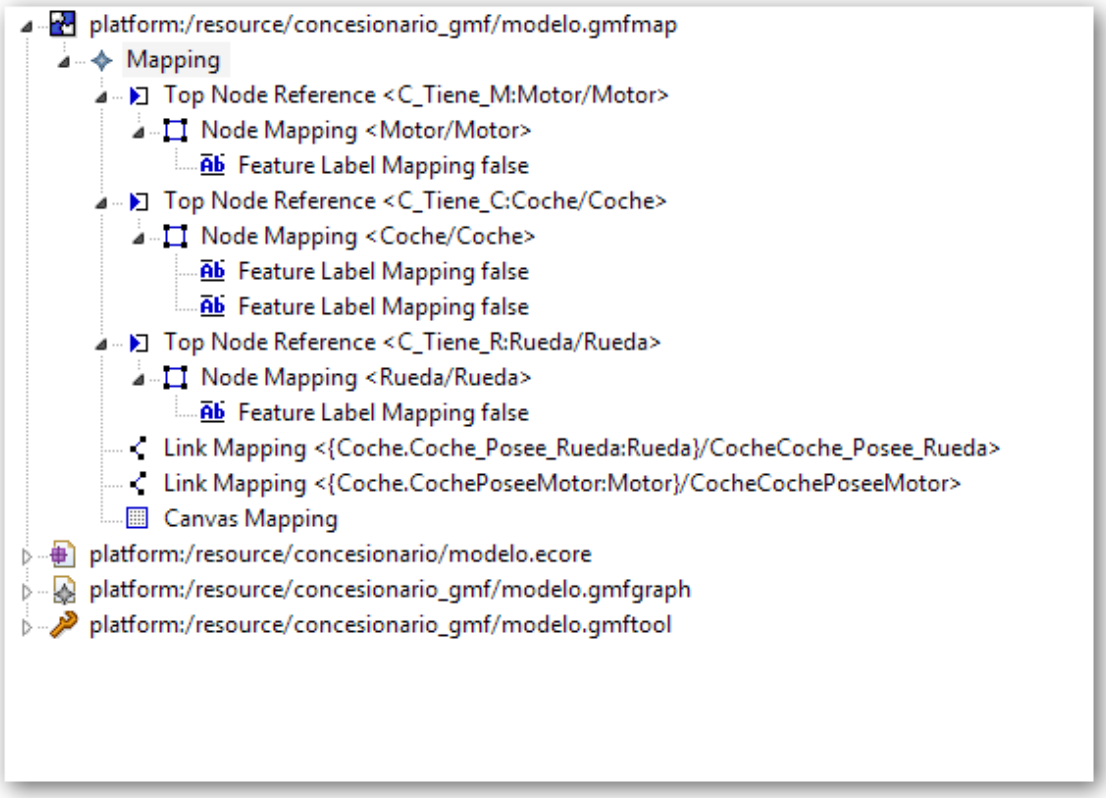


Figura 33: Árbol desplegado del proceso de mapping.

6.8 Generación de código

Una vez acabado el mapping, ya está el proceso de definición de nuestra herramienta completado. Ahora solamente falta generar su código, para poder ser ejecutada. Para poder generar el código, debemos tener un generador. Este elemento nos lo proporciona el *modelo.gmfmap*. Para conseguirlo debemos clicar con el botón derecho sobre la raíz del árbol de *mapping* y seleccionar *Create Generator Model…*.

- Introducimos el nombre del generador, en nuestro caso *modelo.gmfgen*. Pulsamos *Next*.
- Seleccionamos modelo.gmfmap, si no aparece por defecto, pulsamos en *Find in Workspace* para seleccionarlo. Pulsamos *Next*.
- Seleccionamos modelo.genmodel, si no aparece por defecto, pulsamos en *Find in Workspace* para seleccionarlo. Pulsamos *Next*.
- Pulsamos *Finish*.

El explorador de paquetes debe quedar cómo muestra la Figura 34.

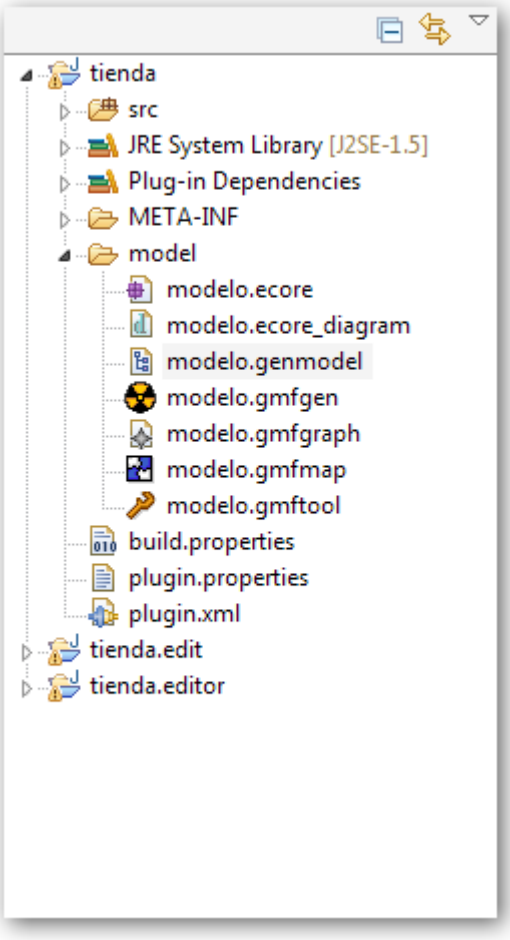


Figura 34: Explorador después de la creación del generador

Finalmente, pulsamos con el botón derecho sobre *modelo.gmfgen* y seleccionamos *Generate Diagram Code*. Si se ha generado correctamente aparecerá un mensaje de confirmación. Se genera una carpeta con el nombre *tienda.diagram*.

6.9 Ejecución de la herramienta de creación de diagramas

En este punto, ya disponemos de nuestra aplicación. Para ejecutarla, hay que seleccionar la carpeta tienda.diagram y pulsar sobre ejecutar (flecha verde de la barra de herramientas). Seleccionar *Run as Eclipse application*.

Se nos abrirá una nueva ventana de Eclipse. Creamos un nuevo proyecto java *File* → *New* → *Java Project*. Lo llamamos *Nuevo* y pulsamos *Finish*.

Ahora pulsamos con el botón derecho sobre *Nuevo* y seleccionamos *New* → *Example* → *Tienda Diagram*. Asignamos los nombres que queramos a los nuevos documentos,

uno será el diagrama y otro el fichero asociado con el lenguaje específico de dominio. Nuestra aplicación debería tener el aspecto de la Figura 35, en la que se ha modelado un coche con un motor y cuatro ruedas.

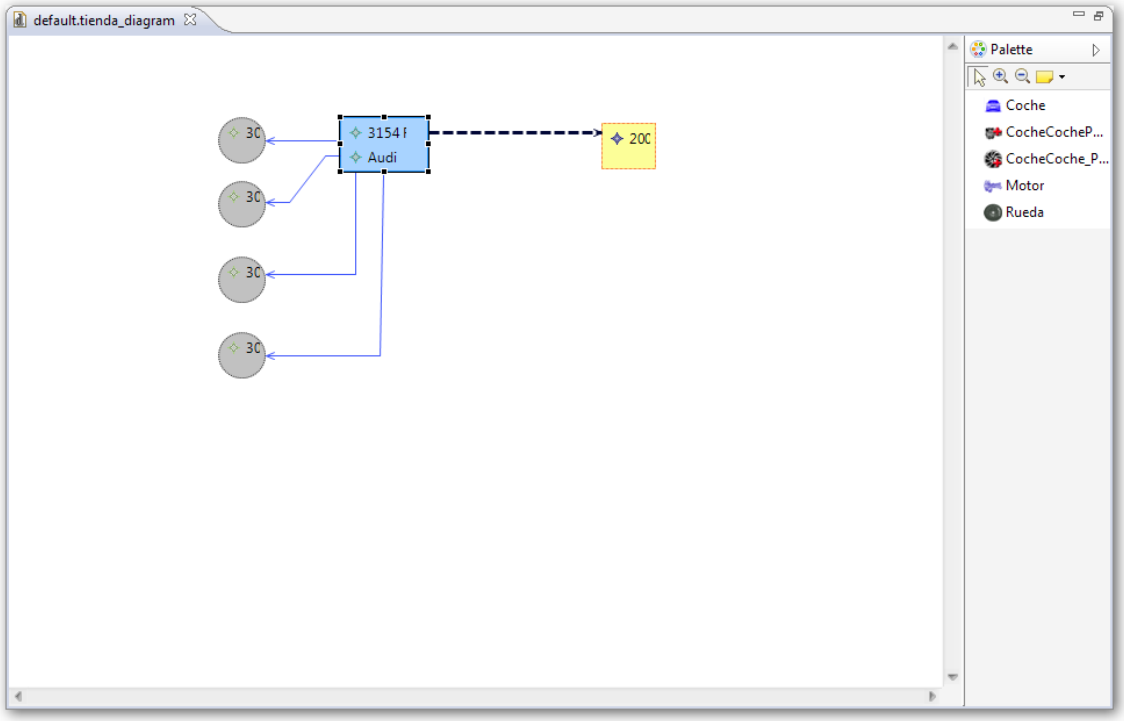


Figura 35: Diagrama Generado

7. CONCLUSIONES

En este documento se han descrito en forma de tutorial los pasos a seguir para crear una herramienta para modelar haciendo uso de un lenguaje específico de dominio. La construcción de la herramienta hace uso de EMF y GMF, herramientas de trabajo proporcionadas por la plataforma de desarrollo Eclipse. A lo largo del tutorial se ha utilizado un caso de estudio sencillo, que permite centrarse en qué momento y cómo se deben ir creando los diferentes documentos y modelos que son necesarios realizar para conseguir crear, finalmente, una herramienta para el modelado de un dominio específico. Este documento corresponde a una versión inicial del tutorial que será mejorado en futuras versiones, y constituye un documento de partida para los alumnos de la asignatura de Técnicas Avanzadas para el Desarrollo de Aplicaciones que

se imparte en las titulaciones de Ingeniería Técnica de Gestión y de Sistemas de la Universidad Politécnica de Madrid.

BIBLIOGRAFÍA

[ECL09] Eclipse. Eclipse - an open development platform. http://www.eclipse.org/

[EMF09] EMF: Eclipse Modeling Framework Project (EMF)

http://www.eclipse.org/modeling/emf/

[GMF09] GMF: The Eclipse Graphical Modeling Framework (GMF),

http://www.eclipse.org/modeling/gmf/